VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV PROCESNÍHO A EKOLOGICKÉHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF PROCESS AND ENVIRONMENTAL ENGINEERING

OPTIMALIZACE KOGENERAČNÍHO SYSTÉMU

OPTIMIZATION OF COGENERATION SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Ing. RADEK STACHA

VEDOUCÍ PRÁCE
SUPERVISOR
Doc. Ing. ZDENĚK JEGLA, Ph.D.

BRNO 2014

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav procesního a ekologického inženýrství
Akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Radek Stacha

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Procesní inženýrství (3909T003)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Optimalizace kogeneračního systému

v anglickém jazyce:

Optimization of cogeneration system

Stručná charakteristika problematiky úkolu:

Předmětem práce je optimalizace modelového případu provedení kogeneračního systému za účelem posouzení výpočtových vlastností vyvíjených optimalizačních metod. Součástí řešení je popis použitých optimalizačních metod a uvedení jejich vývojových algoritmů (tj. blokových schémat). V práci budou porovnány výsledky optimalizace kogeneračního systému získané vlastními vyvíjenými optimalizačními metodami s výsledky optimalizace modelového případu dostupnými z odborné literatury.

Cíle diplomové práce:

a) výchozí popis modelového případu kogeneračního systému a uvedení rovnic jeho matematického modelu;
b) popis vyvíjených optimalizačních metod včetně jejich vývojových algoritmů (blokových schémat);
c) aplikace vyvíjených optimalizačních metod na modelový případ kogeneračního systému;
d) srovnání výsledků optimalizace kogeneračního systému získaných vyvíjenými optimalizačními metodami s výsledky jiných optimalizačních metod dostupných z odborné literatury.

Seznam odborné literatury:

[1] PADILHA, R. S., SANTOS H. F. L., COLACO M. J. and CRUZ M. E., Single and Multi-Objective Optimization of a Cogeneration System Using Hybrid Algorithms, Heat Transfer Engineering, vol. 30, no. 4, pp. 261 – 271, 2009.
[2] TSATSARONIS, G., Invited Papers on Exergoeconomics, Energy, The International Journal, vol. 19, no. 3, pp. 279 – 381, 1994.

Vedoucí diplomové práce: doc. Ing. Zdeněk Jegla, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2013/2014.

V Brně, dne 25.11.2013

L.S.

prof. Ing. Petr Stehlík, CSc., dr. h. c.	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c.
Ředitel ústavu	Děkan fakulty

Abstrakt

Diplomová práce je zaměřena na optimalizaci případu kogeneračního systému za účelem posouzení výpočtových vlastností optimalizačních metod. V práci je uveden popis jednotlivých metod spolu s blokovými diagramy. V první části práce jsou jednotlivé výpočtové algoritmy představeny a navzájem porovnány. Na základě porovnání je následně vytvořen hybridní model, kterým je optimalizován modelový případ kogeneračního systému. Jednotlivé porovnávané algoritmy jsou spolu s vytvořenými hybridními algoritmy součástí příloh.

Klíčová slova: kogenerační systém, optimalizace, genetické algoritmy, optimalizace hejnem částic, hybridní algoritmus, testovací funkce, účelová funkce.

Abstract

Master thesis is focused on optimization of cogeneration system for purpose of rating optimization methods and evaluating properties of these methods. For each method there is description together with block schemes. First part of thesis is devoted to description of methods and their comparison. Second part consists of development of hybrid algorithm, which is used to optimize cogeneration systém model. Each algorithm compared is together with hybrid algorithms included in annexes

Key words: cogeneration systém, optimization, genetic algorithms, particle swarm optimisation, hybrid algorithm, test functions, fitness function

Bibliografická citace

STACHA, R. *Optimalizace kogeneračního systému.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014. 56 s. Vedoucí diplomové práce doc. Ing. Zdeněk Jegla, Ph.D.

Prohlášení o původnosti

Tímto prohlašuji, že jsem předkládanou diplomovou práci vypracoval samostatně,
s využitím uvedené literatury a podkladů, na základě konzultací a pod vedením vedoucího diplomové práce.

V Brně 29.5.2014 …….

Podpis

Poděkování

Tímto děkuji panu doc. Ing. Zdeňkovi Jeglovi, Ph.D. za cenné připomínky a rady týkající se zpracováním diplomové práce.

Obsah

Seznam použitých symbolů

(Pro kapitolu - 1 Kogenerační systém)

Symbol	Význam	Jednotka
$\dot{C}$	Nákladový tok	$/s
$\dot{Q}$	Tepelný tok	W
$\dot{W}$	Práce	MW
$\dot{m}$	Hmotnostní tok	kg/s
c	Cena za jednotku energie	$/J
C	Konstanta v pořizovacích nákladech	
c_p	Měrná tepelná kapacita za konstantního tlaku	kJ/kgK
CRF	Faktor obnovy kapitálu	%
F	Účelová funkce	$/s
h	Měrná entalpie	kJ/kmol
N	Roční fond hodin operující kogenerační jednotky	h
P	Tlak	bar
s	Entropie	J/K
T	Teplota	K
x	Molární poměr	-
Z	Kapitálové náklady na komponentu kogenerační jednotky	$
LHV	Výhřevnost paliva	kJ/kg
M	Molekulová hmotnost	g/mol

Řecké písmena	Význam	Jednotka
γ	Poissonova konstanta	-
η	Účinnost	%
φ	Faktor údržby	-

Dolní index	Význam	Jednotka
0	Referenční prostředí	
a	Vzduch (air)	
$PINCH$	Bod PINCH	
$APPROACH$	Přiblížení	
AC	Vzduchový kompresor	
APH	Předehřívač vzduchu	
CC	Spalovací komora	
f	Palivo pro kogenerační systém	
g	Spaliny	
GT	Plynová turbína	
$HRSG$	Kotel na odpadní teplo	
j	Látka	
l	Tepelná ztráta	
s	Pára	
T	Celkový kogenerační systém	
s	Hřídel	
$isen$	Izoentropický děj	
act	Aktuální děj	

Dolní index	Význam	Jednotka
EC	Ekonomizér	
EV	Výparník	
LM	Střední logaritmický spád	

Úvod

Dnešní návrh a provoz energetických systémů musí zohledňovat efektivní použití přírodních zdrojů, snižovat škody na životním prostředí a zároveň zohledňovat udržitelný rozvoj**.**

Multidisciplinární záběr procesního a ekologického inženýrství dokáže zodpovědět tyto otázky týkající se životního prostředí, stejně jako zohlednit proces tvorby nákladů produktů systému a dokáže pomoci při systémové optimalizaci. V posledních několika dekádách bylo vyvinuto velké množství technik [1] pro analýzu energetických systémů a jejich následnou optimalizaci.

Výzkum, který provedl Valero a kol. [2], je zaměřen na zhodnocení a interpretaci metodologií zabývajících se rozdílnými složkami nákladů, stejně jako exergoekonomickými optimalizacemi. Pro porovnání těchto metodologií byl jako srovnávací příklad (benchmark) použit tzv. **CGAM** problém, což je modelový případ kogeneračního systému formulovaný skupinou specialistů **C**. Frangopulos, **G**. Tsatsaronis, **A**. Valero a **M**. von Spakovsky. Tento modelový problém je sestavený z pěti komponenta a je přijat širokou veřejností pro srovnávání optimalizačních metod.

K optimalizaci tohoto problému je zapotřebí matematických metod, které zohlední termodynamické vlastnosti systému a naleznou optimální řešení z pohledu celkových nákladů (tj. součtu investičních a provozních). Vzrůstá význam robustních, rychlých a účinných matematických optimalizačních modelů resp. algoritmů, jelikož celý proces optimalizace musí být efektivní vůči velkému počtu proměnných, jejíž počet s větší komplexností systému exponenciálně vzrůstá [3].

Cílem diplomové práce je představit některé z nejvíce rozšířených algoritmů pro optimalizaci tohoto typu problému. Tyto algoritmy jsou v práci popsány z hlediska své vnitřní struktury, na základě které je následně napsán pro každý algoritmus výpočtový kód v programu Matlab. Algoritmy jsou následně srovnávány dle své výkonnosti za různých kritérií, na základě které bude vytvořen hybridní model.

Hlavním cílem prováděných činnosti v diplomové práci v oblasti optimalizace kogeneračního systému je vytvoření algoritmu, který má lepší funkčnost na dané oblasti přípustných řešení definovaných soustavou omezujících podmínek, která je dána fyzikálními, termodynamickými a ekonomickými závislostmi kogeneračního systému.

Souvisejícím cílem je představení struktury jednotlivých optimalizačních metod, pro jejich bližší pochopení. Mnoho těchto optimalizačních funkcí má již své místo v různých matematických a statistických programech, nicméně jejich použití může být značně omezeno neznalostí jednotlivých algoritmických struktur a algoritmického chování na určitých typech problémů.

Předložená diplomová práce obsahuje v úvodní části informace o optimalizovaném kogeneračním systému a jeho jednotlivé rovnice, ze kterých se následně vychází při vytvoření optimalizační funkce. Tato účelová funkce je optimalizována algoritmy, které jsou představeny v další části práce.

1 Kogenerační systém

Pojem kogenerace označuje současnou výrobu (generování) více druhů energie, což nejčastěji bývá elektrická a tepelná energie. Systém, který takovou výrobu energií realizuje v určitých parametrech spotřeby, se nazývá kogenerační systém. Trendem v současné době je tzv. trigenerace, kdy dochází ke kombinované výrobě elektřiny, tepla a chladu.

Hlavní výhodou kogeneračního systému je především využití odpadního tepla spalin, při generování elektřiny namísto výroby tepla v odděleném systému. Tímto je automaticky zajištěno snížení produkce emisí a zvyšuje se tepelná účinnost systému, což zlepšuje celkovou efektivitu systému. Kogenerační systémy se dále dělí [4]:

- „Horní“ kogenerační systémy
- „Dolní“ kogenerační systémy

U „horních“ kogeneračních systémů dochází nejdříve k získání energie v příslušné části systému (např. ve spalovací komoře), kde je teplo využito pro technologické procesy a následně odvedeno do zařízení generující elektrickou energii, kterým je např. plynová turbína. Získaná mechanická energie z tepelného motoru je transformována na elektrickou energii v generátorech [4].

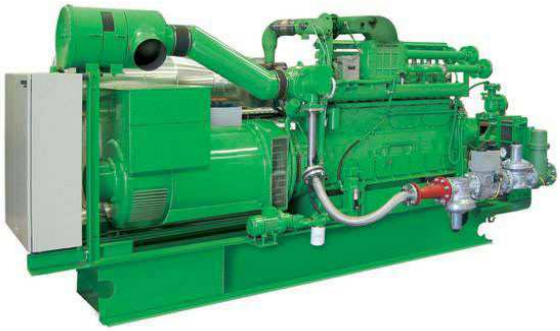
U „dolních“ kogeneračních systémů nejdříve dochází k výrobě elektrické energie, tepelná energie se získává z odváděného tepla v tepelném oběhu. Tyto systémy se využívají více než horní vzhledem k nižší vstupní teplotě do tepelného oběhu [4].

V kogeneračních jednotkách „dolních“ kogeneračních systému se provádí plynulá přeměna primární energie obsažené v palivu na energii elektrickou. Tepelnou energii, kterou už nelze přeměnit na elektrickou, lze využít na dodávku tepla. Technologie v kogeneračních systémech dělíme z fyzikálního hlediska [4]:

- S přímým způsobem přeměny energie
- S nepřímým způsobem přeměny energie

Nepřímý způsob transformace energie se provádí prostřednictvím energetických přeměn. Nejvíce používaný způsob zahrnuje tři transformace. První je uvolnění energie obsažené v palivu nebo regenerace energie z primárního zdroje. Z této energie je získávána mechanická práce, která se používá na mechanický pohon spotřebičů. Tato mechanická energie je nakonec transformována na energii elektrickou. U přímého způsobu transformace se provádí přeměna energie obsažené v palivu přímo na elektrickou energii [4].

Kogenerační jednotky, jaké například ilustruje obr. 1, se mohou skládat ze zařízení pro úpravu primárního zdroje energie, primární jednotky nebo zařízení na výrobu a úpravu elektrické energie, ale i zařízení pro rekuperaci tepelné energie.



Obr. 1 Kogenerační jednotka 2MW [5]

Tepelná energie je rekuperována a odváděna teplonosnými médii, kterými bývá nejčastěji voda, vodní pára, popřípadě vzduch. Parametry kogeneračních jednotek popisují fyzikální, konstrukční, provozní, ekonomické a environmentální veličiny a jejich vzájemné závislosti [6].

1.1 Historický kontext optimalizačního modelu

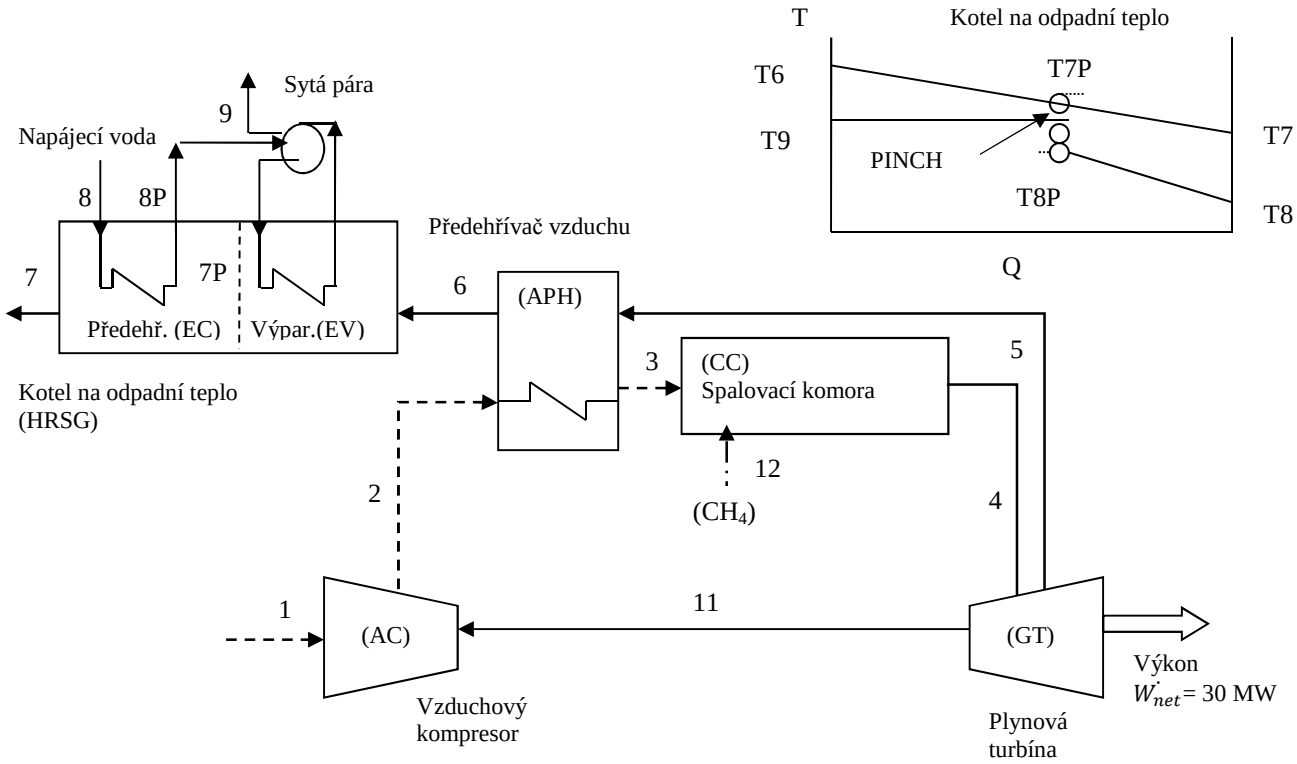
Problém CGAM, který uvedl Valero [2], je model ekonomické optimalizace zjednodušeného kogeneračního systému, který zahrnuje fyzikální, termodynamické a ekonomické modely resp. vazby. Základním předpokladem tohoto modelu je chování spalin jako ideálního plynu a konstantních tepelných kapacit. Tento předpoklad systém zjednodušuje bez výrazné ztráty nebo změny všeobecnosti modelu. Ve všech komponentách, kromě spalovací komory je uvažován adiabatický děj. Jako palivo je uvažován zemní plyn, který je v modelu pro zjednodušení zastoupen metanem.

Tento optimalizační model byl představen na Mezinárodním symposiu „Efficiency, Costs, Optimization and Simulation of Energy Systems (ECOS 92)“ ve Španělské Zaragoze v červnu 1992. Konvenční řešení CGAM problému bylo následně publikováno v časopise Energy Journal. K problému CGAM pak přistoupilo za použití technicko-ekonomických přístupů i přístupů exergo-ekonomických mnoho dalších vědeckých pracovníků, například Padilha, Frangopulos a Tsatsaronis. Tím je v současné době tento problém brán jako modelový případ (benchmark) v oblasti optimalizace kogeneračních systémů

1.2 Konvenční CGAM problém

Model, který je zde uveden bude dále rozveden a popsán na základě literatury [2], jednotlivé rovnice a postupy dále doplňuje Knopf [6]. Model na obr. 2 popisuje kogenerační jednotku o elektrickém výkonu 30 MW, která také generuje 14kgs^{-1} syté páry při tlaku 20 bar. Struktura kogenerační jednotky je zobrazena na obr. 2. Model se skládá ze vzduchového kompresoru (air compressor - AC), předehřívače vzduchu (air preheater - APH), spalovací komory (combustion chamber - CC), plynové turbíny (gas turbine - GT) a kotle na odpadní teplo (heat recovery steam generator – HRSG). Předehřívač vzduchu využívá tepelnou energii ze spalin,

pro ohřev vzduchu vstupujícího do spalovací komory. Kotel na odpadní teplo je složen ze sekce ekonomizéru, kde je předehřívaná vstupní voda, a výparníku, kde je předehřátá voda odpařována. Referenční podmínky jsou definovány jako T_0 = 298.15 K a P_0= 1.013 bar. Palivem pro spalovací komoru je metan resp. zemní plyn s výhřevností (*LHV*) 50 000 kJkg^{-1}



Obr. 2 Schéma modelového kogeneračního systému [3]

Soustava rovnic popisující fyzikální chování systému je popisována v další části práce a je označena jako fyzikální model. Soustava rovnic stavů používaných pro výpočet termodynamických veličin je označena jako termodynamický model a soustava rovnic pro výpočet investičních a provozních nákladů jednotlivých komponent systému je označena jako ekonomický model.

Stavové proměnné systému vybrané pro optimalizaci jsou:

- poměr tlaků P_2/P_1;
- izoentropické účinnosti vzduchového kompresoru (η_{AC});
- spalovací turbíny (η_{GT});
- teploty vzduchu při výstupu z předehřívače vzduchu (T_3);
- teplota spalovacího vzduchu ve spalovací turbíně (T_4);

Optimalizační model, jehož dílčí submodely budou popsány v další části práce, je založen na následujícím výčtu jednotlivých zjednodušujících předpokladů řešení:

- Vzduch a spaliny se chovají jako ideální plyn s konstantními tepelnými kapacitami
- Zemní plyn je zde uvažován jako čistý metan CH4

- Ve všech komponentech, kromě spalovací komory, probíhá adiabatický děj
- Změny tlaků pro vzduch (ΔP_{CC}) a pro toky spalin ve spalovací komoře, předehřívači vzduchu (ΔP_{APH}) a HRSG (ΔP_{HRSG}) jsou dány.

1.2.1 Fyzikální model

Fyzikální model je tvořen soustavou rovnic pro hmotnostní a energetickou bilanci každé z komponent kogeneračního systému. Jedná se o standartní rovnice, které lze nalézt v učebnicích procesního inženýrství.

Vzduchový kompresor (AC)

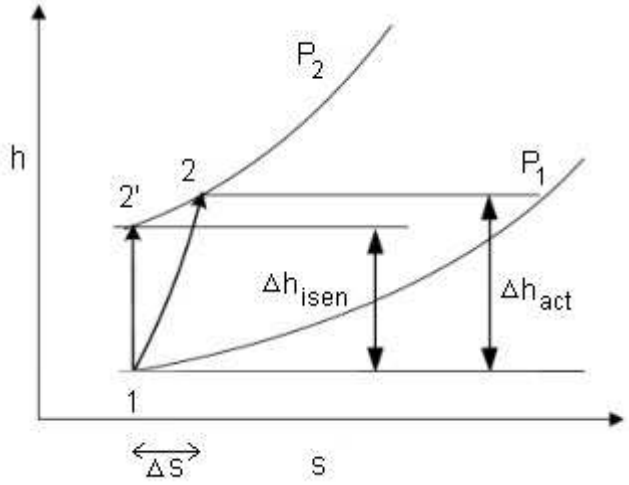
U vzduchového kompresoru dochází ke stlačení ideálního plynu při zvýšení tlaku z P_1 na P_2. To má za následek zvýšení teploty z T_1 na T_2. Pro systém s konstantními tepelnými kapacitami platí:

$$h_2 - h_1 = \Delta h = c_{p,g}(T_2 - T_1) \tag{1}$$

Vzhledem k předpokladu adiabatického děje je výměna tepla mezi kompresorem a okolím zanedbatelná, a proto je v rovnici (2) člen $dQ/dt=0$.

$$\frac{dQ}{dt} - \frac{dW_s}{dt} = \dot{m}(\Delta h) \tag{2}$$

Kompresor je poháněn turbínou, jak uvádí obr. 2. Práce, kterou koná hřídel, pochází z okolí a je tedy záporná. Vzhledem k záporné hodnotě práce je pak v kompresním procesu $T_2 > T_1$.



Obr. 3 Graf závislosti entalpie-entropie [6]

Na obr. 3 lze vidět dva procesy. Prvním z nich je proces izoentropický, reversibilní, kdy změna ze stavu 1 do 2‘ je rovna změně entalpie Δh_{isen}. Druhým pohybem ze stavu 1 do stavu

2 je proces, který opravdu nastane s ohledem na účinnost vzduchového kompresoru. Ta se promítne do práce, kterou vzduchový kompresor vykoná, jak uvádí rovnice (3)

$$\frac{dW_{s,act}}{dt} = \frac{\frac{dW_{s,isen}}{dt}}{\eta_{AC}} \tag{3}$$

$$c_{p,a}(T_2 - T_1) = \frac{c_{p,a}\left(T_1\left(\frac{P_2}{P_1}\right)^{\frac{\gamma_a - 1}{\gamma_a}} - T_1\right)}{\eta_{AC}} \tag{4}$$

Rovnici (4) upravíme na rovnici (5) pro získání teploty T_2

$$T_2 = T_1\left\{1 + \frac{1}{\eta_{AC}}\left[\left(\frac{P_2}{P_1}\right)^{\frac{\gamma_a - 1}{\gamma_a}} - 1\right]\right\} \tag{5}$$

$$\dot{W}_{AC} = \dot{m}_a c_{p,a}(T_2 - T_1) \tag{6}$$

Kde T_1 je okolní teplota vzduchu; P_1 je tlak okolí; T_2 je teplota vzduchu na výstupu z kompresoru (AC); η_{AC} je isoentropicka účinnost kompresoru (AC); γ_a je Poissonova konstanta. $\dot{W}_{AC}$ je práce kompresoru (AC), $\dot{m}_a$ je hmotnostní průtok vzduchu a $c_{P.a}$ je tepelná kapacita vzduchu.

Předehřívač vzduchu (APH)

V předehřívači vzduchu dochází k výměně tepla mezi spalinami, které jsou odváděny do kotle na odpadní teplo (HRSG) a vzduchem o teplotě T_2 a tlaku P_2. Vzhledem k podmínce adiabatického děje, nedochází k výměně tepla s okolím, proto lze rovnici tepelné bilance předehřívače zapsat ve tvaru:

$$\dot{m}_a c_{p,a}(T_3 - T_2) = \dot{m}_g c_{p,g}(T_5 - T_6) \tag{7}$$

Procentuální pokles tlaku pracovních látek (tj. spalin a vzduchu), který nastává v předehřívači je brán v potaz rovnicemi (8) a (9), které upravují tlaky vystupující z předehřívače na straně spalin i na straně předehřátého vzduchu.

$$P_3 = P_2(1 - \Delta P_{a,APH}) \tag{8}$$

$$P_6 = P_5(1 - \Delta P_{g,APH}) \tag{9}$$

Kde je $\dot{m}_g$ hmotnostní tok spalin; $c_{P.g}$ je tepelná kapacita spalin; T_3 je teplota vzduchu na výstupu z předehřívače (APH); T_5 je teplota spalin na výstupu z turbíny; T_6 je teplota spalin na výstupu z předehřívače vzduchu; P_3 je tlak na výstupu z předehřívače (APH); $\Delta P_{a,APH}$ je procentuální pokles tlaku na straně vzduchu v předehřívači vzduchu (5%); P_5 je tlak spalin vystupujících z turbíny; P_6 je tlak spalin vystupujících z předehřívače vzduchu a $\Delta P_{g,APH}$ je procentuální pokles spalin na straně předehřívače (3%).

Spalovací komora (CC)

Ve spalovací komoře dochází ke spálení směsi vzduchu a zemního plynu. Je uvažováno s konstantními hmotnostními toky vzduchu tak zemního plynu. Zahrnutí průtoku paliva do hmotnostního toku spalin, je provedeno hmotnostní bilancí dle rovnice (10):

$$\dot{m}_g = \dot{m}_a + \dot{m}_f \tag{10}$$

$$\dot{m}_a h_3 + \dot{m}_f LHV = \dot{m}_g h_4 + \dot{Q}_{1,CC} \tag{11}$$

$$h_3 = c_{p,a}(T_3 - T_0) \tag{12}$$

$$h_4 = c_{p,g}(T_4 - T_0) \tag{13}$$

$$\dot{Q}_{1,CC} = \dot{m}_f LHV(1 - \eta_{CC}) \tag{14}$$

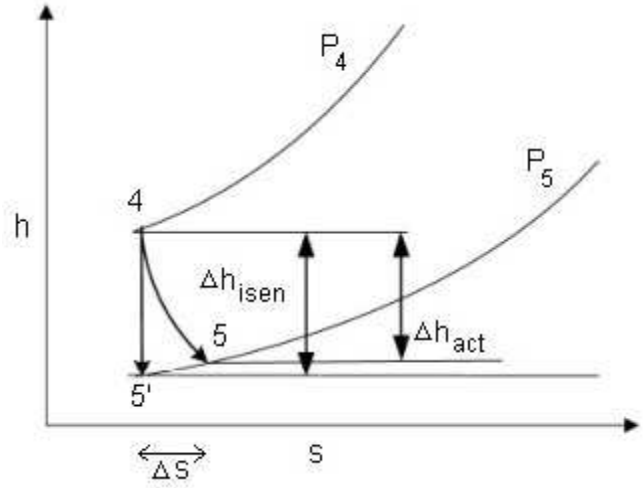
$$P_4 = P_3(1 - \Delta P_{CC}) \tag{15}$$

Kde $\dot{m}_f$ je hmotnostní průtok paliva; T_4 je teplota spalin vystupující ze spalovací komory; $\dot{Q}_{1,CC}$ je tepelná ztráta ve spalovací komoře (CC); η_{CC} je tepelná účinnost spalovací komory (CC); P_4 je tlak při výstupu ze spalovací komory (CC); ΔP_{CC} je procentuální ztráta tlaku spalin uvnitř spalovací komory

Plynová turbína (GT)

Teplo, které bylo uvolněno spalováním paliva ve spalovací komoře, je transformováno na práci, kterou vykoná hřídel turbíny. Spaliny vycházející ze spalovací komory expandují při snížení tlaku z P_4 na P_5 a to má za následek snížení teploty z T_3 na hodnotu T_4.

Obr. 4 ukazuje, že podobně jako u kompresoru (AC), nebude docházet ke změně stavu z hodnoty entalpie v bodě 3 do bodu 4‘, ale vzhledem k účinnosti turbíny proběhne reálně změna ze stavu 3 do stavu 4.



Obr. 4 Graf závislosti entalpie-entropie [6]

Vzhledem k tomu, že práce hřídele turbíny je dána spalovacím procesem, pak vzhledem k rovnici (2) je práce, kterou vykoná turbína, kladná ve své hodnotě. Z toho lze vyvodit, že $T_5 < T_4$.

$$T_5 = T_4 \left\{ 1 - \eta_{GT} \left[1 - \left(\frac{P_4}{P_5} \right)^{\frac{1-\gamma_g}{\gamma_a}} \right] \right\} \tag{16}$$

$$\dot{W}_{GT} = \dot{m}_g c_{p,g} (T_4 - T_5) \tag{17}$$

$$\dot{W}_{net} = \dot{W}_{GT} - \dot{W}_{AC} \tag{18}$$

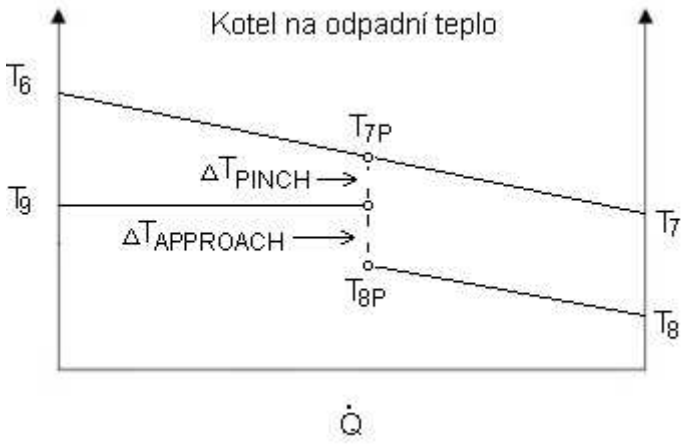
η_{GT} je izoentropická účinnost plynové turbíny (GT); γ_g je Poissonova konstanta pro spaliny; $\dot{W}_{GT}$ je práce plynové turbíny a $\dot{W}_{net}$ je práce celého systému.

Kotel na odpadní teplo (HRSG)

V kogeneračním systému dochází k tvorbě páry a elektrické energie. K výrobě elektrické energie v systému dochází v plynové turbíně. K výrobě páry dochází v HRSG. Zde dochází k recyklaci tepelné energie, která zůstává ve spalinách.

Kondenzovaná voda ze systému je vedena zpět do HRSG, kde je nejprve přivedena blízko bodu varu v části zvané ekonomizér (předehřívač) a následně je transformována na páru v druhé části nazývané výparník.

V HRSG jsou sledovány dva důležité parametry. Teplotní rozdíl ΔT_{PINCH} a teplotní rozdíl $\Delta T_{APPROACH}$. V obr. 5 můžeme vidět, že voda při teplotě T_8 a tlaku P_8 je nejprve ohřáta na teplotu o $\Delta T_{APPROACH}$ menší, než je bod varu, a poté přeměněna na páru o teplotě T_9 a tlaku P_9.



Obr. 5 Kotel na odpadní teplo **[6]**

V HRSG jsou teploty ΔT_{PINCH} a $\Delta T_{APPROACH}$ a T_7, P_7 a T_8 P_8 důležité, aby například v ekonomizéru nedocházelo ke korozi teplosměnných ploch vzhledem k teplotám spalin v blízkosti rosného bodu. Zvyšování hodnoty ΔT_{PINCH} znamená klesající možnost výměny tepla ze spalin.

Malá hodnota $\Delta T_{APPROACH}$ zase způsobí, že k vypařování bude docházet už v ekonomizéru.

Rovnice pro tepelnou bilanci kotle na odpadní teplo:

$$\dot{Q}_{evap,g} = \dot{m}_g(h_{7P} - h_6) \tag{19}$$

$$T_{8P} = T_9 - \Delta T_A \tag{20}$$

$$\Delta T_P = T_{7P} - T_9 > 0 \tag{21}$$

$$\dot{m}_g c_{p,g}(T_6 - T_{7P}) = \dot{m}_s(h_9 - h_{8P}) \tag{22}$$

V kogeneračním systému je pracováno s případem ideálního plynu, nicméně pro vodu je důležité vzít reálné hodnoty k výpočtu teploty syté páry.

$$\dot{Q}_{EC,w} = \dot{m}_s(h_{8P} - h_8) \tag{23}$$

$$T_7 = T_6 - \frac{\dot{m}_s(h_9 - h_8)}{\dot{m}_g c_{p,g}} \tag{24}$$

$$P_O = P_6(1 - \Delta P_{HRSG}) \tag{25}$$

T_{8P} je teplota vody na vstupu do výparníku (EV); T_9 je teplota nasycené páry na výstupu z HRSG; ΔT_A je minimální rozdíl teplot; $\dot{m}_s$ je průtok páry; $(h_9 - h_8)$ je entalpický rozdíl na straně voda a pára; ΔT_P je minimální rozdíl teplot na PINCHI; T_{7P} je teplota spalin vystupující z výparníku (EV); $(h_9 - h_{8P})$ je celkový entalpický rozdíl na straně voda a pára; ΔP_{HRSG} je procentuální pokles tlaku spalin v HRSG (5%).

1.2.2 Termodynamický model

CGAM problém byl prezentován v literatuře [2] s následující formulací termodynamického modelu, který je založen na obecných předpokladech standardních inženýrských problémů:

- Vlhkost vzduchu 60%;
- Molové poměry x_{O2} =0,2059, x_{N2}=0,7748, x_{CO2} = 0,0003 a x_{H20} = 0,19;
- Teplota T_0 = 298,15 K;
- Referenční tlak P_0 = 1,013 bar;

Palivo tvoří zemní plyn, který jak bylo dříve zmíněno, je zde nahrazen čistým metanem. Ve spalovací komoře je předpokládáno dokonalé spalování a v rovnici (26) je metan omezujícím reakčním činidlem [2].

$$fCH_4 + x^0_{O_2}O_2 + x^0_{N_2}N_2 + x^0_{CO_2}CO_2 + x^0_{H_2O}H_2O \rightarrow \\ \rightarrow (f + x^0_{CO_2})CO_2 + (f + x^0_{H_2O})H_2O + (x^0_{O_2} - 2f)O_2 + x^0_{N_2}N_2 \quad (26)$$

Dusík a oxid uhličitý na vstupu jsou inertní. Molekulové hmotnosti jsou pro metan M_{CH4} = 16,043 gmol^{-1} a molekulová hmotnost pro vzduch M_a = 28,648 gmol^{-1}.

1.2.3 Ekonomický model

Standardní formulace ekonomického modelu pro řešení CGAM problému je dle [2] založena na vztazích uvedených v této kapitole.

Pořizovací a provozní náklady

Při posouzení celkových převedených nákladů na kogenerační jednotku je nutné zohlednit náklady pořizovací, provozní i náklady na údržby. Vztahy pro vyjádření pořizovacích nákladů jsou zobrazeny v tab. 1 a doplňující konstanty v tab. 2.

Tab. 1 Rovnice pořizovacích nákladů **[2]**

Kompresor	$\dot{Z}_{AC} = \left(\frac{C_{11}\dot{m}_a}{C_{11} - \eta_{AC}}\right)\left(\frac{P_2}{P_1}\right) ln\left(\frac{P_2}{P_1}\right)$
Spalovací komora	$\dot{Z}_{CC} = \left(\frac{C_{21}\dot{m}_a}{C_{22} - \frac{P_4}{P_3}}\right)(1 + EXP(C_{23}T_4 - C_{24}))$
Plynová turbína	$\dot{Z}_{GT} = \left(\frac{C_{31}\dot{m}_g}{C_{32} - \eta_{GT}}\right) \ln\left(\frac{P_4}{P_5}\right)(1 + EXP(C_{33}T_4 - C_{34}))$
Předehřívač vzduchu	$\dot{Z}_{GT} = \left(\frac{\dot{m}_g(h_5 - h_6)}{U\Delta T_{LM,APH}}\right)^{0,6} \quad \Delta T_{LM,APH} = \frac{(T_5 - T_3) - (T_6 - T_2)}{ln\left(\frac{T_5 - T_3}{T_6 - T_2}\right)}$
HRSG	$\dot{Z}_{GT} = \left[\left(\frac{\dot{Q}_{EC}}{\Delta T_{LM,EC}}\right)^{0,8} + \left(\frac{\dot{Q}_{EV}}{\Delta T_{LM,EV}}\right)^{0,8}\right] + C_{52}\dot{m}_s + C_{53}{\dot{m}_g}^{1,2}$ $\Delta T_{LM,EV} = \frac{(T_6 - T_9) - (T_{7P} - T_{8P})}{ln\left(\frac{T_6 - T_9}{T_{7P} - T_{8P}}\right)}$ $\Delta T_{LM,EC} = \frac{(T_7 - T_8) - (T_{7P} - T_{8P})}{ln\left(\frac{T_7 - T_8}{T_{7P} - T_{8P}}\right)}$

Tab. 2 Konstanty zvolené pro rovnice pořizovacích nákladů **[2]**

Kompresor	$C_{11} = 39{,}5\ \$/kgs^{-1} \quad C_{12} = 0{,}9$
Spalovací komora	$C_{21} = 25{,}6\ \$/kgs^{-1} \quad C_{22} = 0{,}995$ $C_{23} = 0{,}018\ K^{-1} \quad C_{24} = 54{,}4$
Plynová turbína	$C_{31} = 25{,}6\ \$/kgs^{-1} \quad C_{32} = 0{,}92$ $C_{33} = 0{,}036\ K^{-1} \quad C_{34} = 54{,}4$
Předehřívač vzduchu	$C_{41} = 2290\ \$/kgm^{-1,2} \quad U = 0{,}018\ kWm^{-2}K^{-1}$
HRSG	$C_{51} = 3650\ \$/(kWK^{-1})^{0,8}$ $C_{52} = 11820\ \$/kgs^{-1} \quad C_{53} = 658\ \$/(kgs^{-1})^{1,2}$

V tab. 1 je u předehřívače vzduchu $\Delta T_{LM,APH}$ střední logaritmický spád teplot předehřívače; T_2 je teplota vzduchu na vstupu do předehřívače; T_3 je teplota vzduchu na výstupu z předehřívače; T_5 je teplota spalin na vstupu do předehřívače a T_6 je teplota spalin na výstupu z předehřívače. U HRSG je $\Delta T_{LM,EV}$ střední logaritmický spád teplot výparníku; T_6 je teplota spalin na vstupu do HRSG; $T_{7P} - T_9$ udává velikost PINCHE;

Na základě těchto nákladů je následně rovnice pro výpočet převedených pořizovacích (investičních) nákladů včetně nákladů na údržbu dané komponenty zařízení dána vztahem (27).

$$\dot{Z}_i = \frac{Z_i CRF \varphi}{(3600N)} \tag{27}$$

Kde Z_i je pořizovací cena dané komponenty; CRF je faktor roční návratnosti; N reprezentuje počet hodin, kdy je jednotka v provozu a φ je faktor údržby (určuje se φ = 1,06).

Provozní a celkové náklady

Provozní náklady jsou vyjádřeny ve vztahu ke spotřebě paliva rovnicí:

$$\dot{C}_f = c_f \dot{m}_f LHV \tag{28}$$

Celkové náklady potom získáme sumou dílčích nákladů, pomocí vztahu:

$$\dot{C}_T = c_f \dot{m}_f LHV + \sum_{i=1}^{5} \dot{Z}_i \tag{29}$$

Kde c_f je cena paliva za energetickou jednotku; $\dot{m}_f$ je hmotnostní průtok paliva.

1.2.4 Účelová funkce

Fyzikální modely spolu s modely nákladů kogeneračního systému mají pět stupňů volnosti, které jsou reprezentovány vybranými stavovými proměnnými. Optimalizační problém sestává z minimalizace účelové funkce za předpokladu pevné ceny elektrické energie a produkce páry.

Na základě toho pak může být optimalizační problém vyjádřen jako minimalizace funkce $\dot{C}_T$, na oblasti přípustných řešení, jež je formulován fyzikálním, termodynamickým a ekonomickým modelem.

Rovnice účelové funkce po rozepsání rovnice (29) na tvar:

$$\dot{C}_T = c_f \dot{m}_f LHV + \dot{Z}_{AC} + \dot{Z}_{APH} + \dot{Z}_{CC} + \dot{Z}_{GT} + \dot{Z}_{HRSG} \tag{30}$$

Optimalizovaný model je nelineární. Hodnoty optimálního řešení tohoto CGAM problému nalezené autory práce jsou uvedeny v tab. 3. Tyto hodnoty byly nalezeny konvenčními optimalizačními metodami. Toto řešení lze však z praktického pohledu vyhodnotit jako nereálné vzhledem k velikosti minimálního rozdílu teplot na PINCHI pouhých 1.64 K, jak je uvedeno v tab. 4. I přes toto nereálné číslo je model v uvedené podobě využíván pro matematické optimalizace.

Tab. 3 Hodnoty termodynamických veličin CGAM problému

Proměnná	Hodnota
P_2/P_1	8,5234
η_{AC}	0,8468
T_3	914,28 K
η_{GT}	0,8786
T_4	1492,63 K

Tab, 4 Výchozí optimální hodnoty kogeneračního systému

i	T_i (K)	P_i (bar)
1	298,15	1,013
2	595,51	8,634
3	914,28	8,202
4	1492,63	7,792
5	987,90	1,099
6	718,76	1,066
7	400,26	1,013
8	298,15	20,000
9	485,52	20,000

Proměnná	Hodnota
$\dot{m}_a$	99,4559 kgs^{-1}
$\dot{m}_f$	1,6274 kgs^{-1}
ΔT_{PINCH}	1,64 K
$\dot{W}_{GT}$	29692,5 kW
$\dot{W}_{AC}$	59692,5 kW

Autoři jednotlivých studií, kteří se zabývali CGAM problémem a jeho nákladovou funkcí [7], [8], [3], [9], však nedospěli ke shodným závěrům s ohledem na hodnoty jeho termodynamických veličin.

Tabulka 5 ukazuje dílčí výsledky podle rozdílných přístupů, kde dle [3] HA1 a HA2 byla optimalizační metoda pomoci hybridních genetických algoritmů s různými parametry a dle [9] pak V1 a V2 byla optimalizační metoda na základě exergoekonomického systematického přístupu.

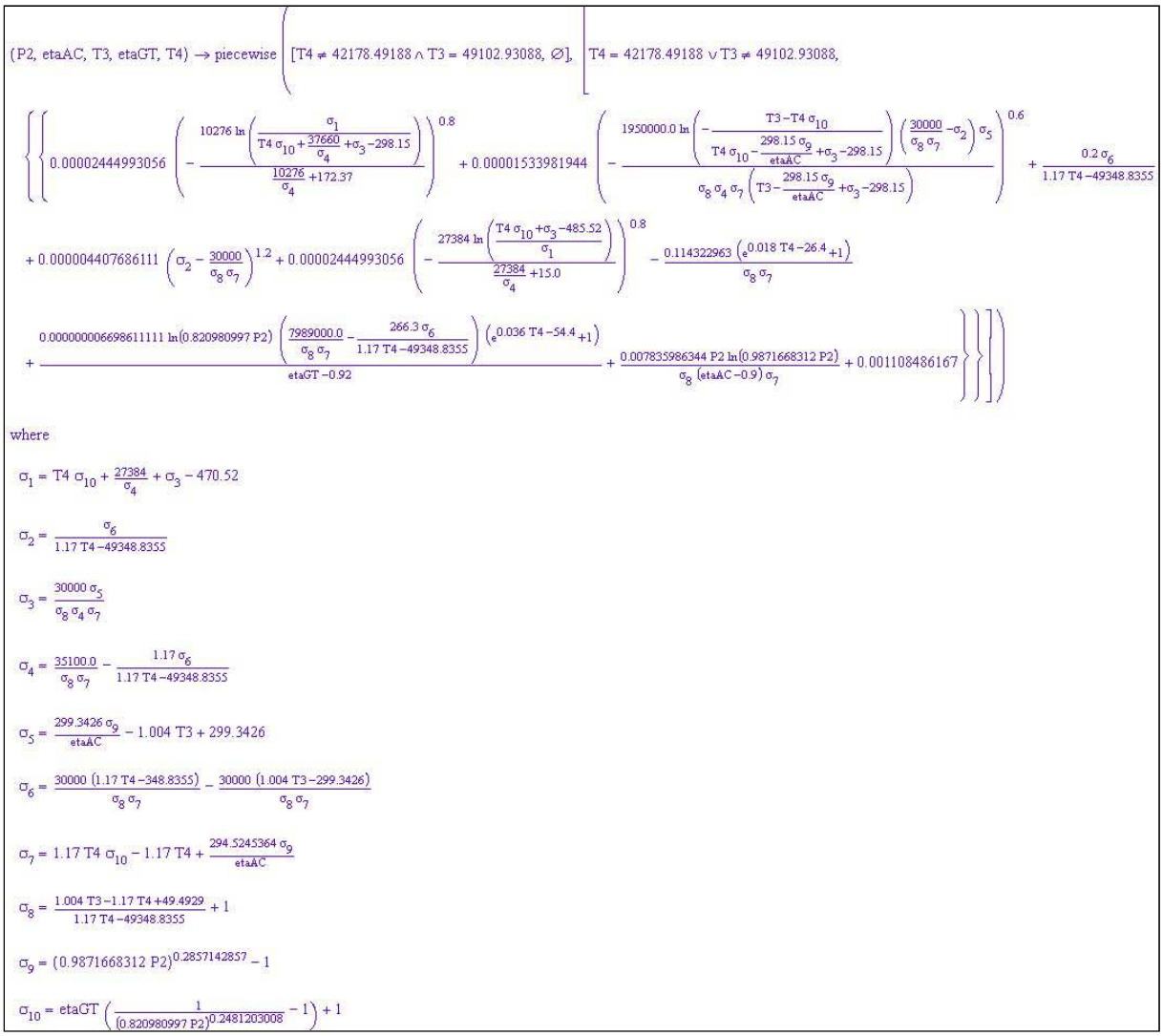
V tabulce 5 lze vidět, že přístup optimalizační metody za pomoci stochastických algoritmů se například v poměru tlaků a v teplotách blížil více řešení CGAM než iterativně náročnější přístup exergoekonomický, kde poměry tlaků zůstaly na celých hodnotách podobně jako teploty.

Odvození účelové funkce bylo provedeno za použití výpočetního programu Matlab a Mupad, který tvoří prostředí pro symbolické výpočty v Matlabu. Jednotlivé rovnice uvedené ve fyzikálním, termodynamickém a ekonomickém modelu byly vyhodnoceny a vyjádřeny

stavovými proměnnými. Účelová funkce, která vychází z rovnice (30) po dosazení následně nabyla tvaru, který pro ilustraci uvádí obr. 6.

Tab. 5 Komparativní hodnoty CGAM problému z jiných studií [2], [3], [9]

Proměnná	CGAM	HA1	HA2	V1	V2
P_2/P_1	8,5234	8,9073	8,3370	10	8
η_{AC}	0,8468	0,8434	0,8436	0,85	0,85
T_3	914,28 K	908,46 K	916,23 K	900 K	933 K
η_{GT}	0,8786	0,8802	0,8781	0,875	0,87
T_4	1492,63 K	1487,94 K	1487,94 K	1485 K	1500 K

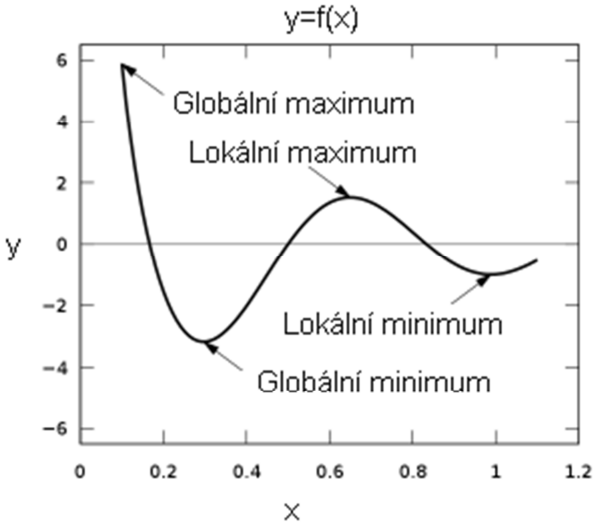


Obr. 6 Ukázka účelové funkce (zdroj Mupad/Matlab)

Účelová funkce byla po symbolickém vyjádření v programu Mupad dále převedena do Matlabu, kde mohla být řešena optimalizačními matematickými modely. Tento výsledný soubor, který uvádí účelovou funkci připravenou pro následné řešení, tvoří Přílohu 1 této práce.

2 Optimalizace

Optimalizace se zabývá nalezením souřadnic takového bodu v definičním oboru funkce, jehož funkční hodnota je extrémní hodnotou dané funkce [10]. Problém nalezení extrému funkce s jedním argumentem ukazuje obr. 7.



Obr. 7 Ilustrace globálních a lokálních extrémů funkce

Globální extrém je minimum nebo maximum na množině všech možných kombinací hodnot jednotlivých stavových proměnných. Lokální extrém je minimum nebo maximum na podmnožině všech možných kombinací hodnot jednotlivých stavových proměnných. Funkce má lokální minima, z nichž jedno lokální minimum tvoří extrém funkce a tedy globální minimum. V bodě globálního minima je funkční hodnota účelové funkce nejmenší v daném oboru hodnot [1].

Úloha funkce bez omezení je představována dle vztahů (31) a (32) takto [11]:

$$f(\boldsymbol{x}), x = (x_1, x_2, x_3, \dots, x_d) \qquad x \in D \tag{31}$$

Lokální a globální minima a maxima spolu s lokálními a globálními vázanými minimy a maximy lze matematicky definovat podle následujících definic, na které odkazuje literatura [11]. Nejprve bude definováno okolí bodu a následně lokální extrémy funkce.

Pro okolí bodu platí [11]:

Buď $x, y \in \mathbb{R}^n$. Potom číslo $d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ se nazývá vzdálenost bodů *x*, *y*. Buď $x_0 \in \mathbb{R}^n, \delta > 0, \delta \in \mathbb{R}$. Pak množina $K(x_0, \delta) = (x_0 \in \mathbb{R}^n, d(x,y) < \delta)$ se nazývá δ-okolí bodu.

Pro lokální maximum pak platí [11]:

$$\text{Když } \exists\, K(x_0, \delta) \subseteq Df \text{ tak, že } \forall x \in K(x_0, \delta) \text{ platí } f(x) \leq f(a) \tag{32}$$

Pro lokální minima pak platí [11]:

$$\text{Když } \exists\, K(x_0, \delta) \subseteq Df \text{ tak, že } \forall x \in K(x_0, \delta) \text{ platí } f(x) \geq f(a) \tag{33}$$

Účelová funkce je funkce udávající kvalitativní kritérium pro hodnocení stavu zkoumaného prostoru. Můžou to být tedy například celkové náklady, hmotnost a podobně. Stavová proměnná je proměnná vyskytující se v účelové funkci

Matematická analýza nabízí různé způsoby, jak nalézt extrémy funkcí, u kterých známe první a druhou derivaci. Účelová funkce však může být, a v praxi většinou je, velmi komplexní a má mnoho lokálních minim. V praxi se pak často stává, že argumentem funkce ani nemusí být jedno reálné číslo ale vektor reálných čísel. Úloha také nemusí být v daném bodě diferencovatelná a mění se tzv. skokově.

Úloha nalezení globálního minima lze formulovat pomocí rovnice (34) takto [11]:

$$f: D \to \mathbb{R}, D \subseteq \mathbb{R}^d \tag{34}$$

Hledáme bod $\boldsymbol{x}^* \in D$, pro který platí, že $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$, pro $\forall \boldsymbol{x}, \boldsymbol{x} \in D$. Nalezení bodu $\boldsymbol{x}^* \in D$ je řešením problému globální optimalizace, kterému se říká globální minimum. Definičnímu oboru D se říká prohledávaný prostor [11].

Globální maximum lze nalézt dle [1] stejně jako globální minimum s tím rozdílem, že u globálního maxima se hledá minimum $g(\boldsymbol{x}) = -f(\boldsymbol{x})$.

Úlohy globální optimalizace je nutné řešit v mnoha praktických problémech a s velmi významným ekonomickým efektem. Algoritmy, které dokáží takovéto problémy řešit efektivně, jsou tudíž důležité.

Úloha optimalizace se označuje jako úloha hledání volného extrému funkce, pokud nejsou požadovány žádné doplňující podmínky. Pokud v úloze vystupují podmínky splnění, pak jde o řešení vázaného extrému.

Úloha funkce s omezeními je představována dle vztahů (31) a (32) takto [11]:

$$f(\boldsymbol{x}), x = (x_1, x_2, x_3, \dots, x_d) \qquad x \in D \tag{35}$$

Podmínky:

$$\begin{aligned} g_i(\boldsymbol{x}) &\leq 0, \quad i = 1, \dots, p \\ h_j(\boldsymbol{x}) &= 0, \quad j = p+1, \dots, m \end{aligned} \tag{36}$$

Řešení je přijatelné pokud $g_i(\boldsymbol{x}) \leq 0$ pro $i = 1, .., p$ a $h_j(\boldsymbol{x}) = 0, \ j = p+1, \dots, m$. Pro libovolný bod $\boldsymbol{x} \in D$.

Pro vázaná lokální minima a maxima pak matematicky platí podobně jako pro volné lokální extrémy:

Pro vázané lokální maximum platí [11]:

$$\begin{gathered}\text{Když } \exists x_0 \in Df \cap V \text{ a } \exists K(x_0, \delta) \\ \text{tak, že } \forall x \in K(x_0, \delta) \cap Df \cap V \text{ platí } f(x) \geq f(a)\end{gathered} \tag{37}$$

Pro vázané lokální minimum pak platí [11]:

$$\begin{gathered}\text{Když } \exists x_0 \in Df \cap V \text{ a } \exists K(x_0, \delta) \\ \text{tak, že } \forall x \in K(x_0, \delta) \cap Df \cap V \text{ platí } f(x) \leq f(a)\end{gathered} \tag{38}$$

Další oblastí jsou problémy, kdy vyhledávaná oblast není spojitá a řešení se mění skokově. Taková funkce není diferencovatelná [12], ale i u takové účelové funkce lze nalézt její extrém. V praxi se takové účelové funkce často vyskytují vzhledem k různým podmínkám, které procesy vyžadují. V diplomové práci jsou proto použity algoritmy, které dokáží řešit skokové i nediferencovatelné funkce.

V diplomové práci půjde o řešení vázaného extrému účelové funkce, kde bude problém řešen v souvislé oblasti. Účelovou funkci lze vyhodnotit na požadovanou přesnost v každém jejím bodě souvislého prostoru řešení.

Při tvoření a řešení účelových funkcí speciálními algoritmy je důležité mít na paměti, že výsledkem hybridního algoritmu může být stejná výkonnost jako u algoritmu předchozích, to je dáno tzv. No free lunch teorémem.

2.1 No free lunch theorem

Teorém žádného obědu zdarma, který definoval Wolpert [13], říká, že všechny algoritmy, které hledají extrém účelové funkce, se chovají stejně, pokud zprůměrujeme výsledky pro všechny možné účelové funkce. V případě že tedy algoritmus A převyšuje svojí výkonností algoritmus B pro určitý počet účelových funkcí, pak podle teorému musí existovat stejný počet jiných účelových funkcí, na kterých algoritmus B převyšuje svou výkonností algoritmus A. Podle teorému pak neexistuje jeden specifický algoritmus, který dokáže převyšovat ostatní na všech typech účelových funkcí.

Metafora no free lunch vychází z příkladu restaurací a jídelních lístků. Každá restaurace reprezentuje algoritmus řešící nějaký druh problému. Tyto restaurace mají jídelní lístek, který je asociován talířem (ve významu problému) s určitou cenou na daném jídelním lístku. Tato cena reprezentuje výkonnost dané procedury řešící problém. Jídelní lístky restaurací jsou identické až na fakt, že ceny jednotlivých jídel jsou navzájem přeházené.

Pro všežravce, který má stejnou pravděpodobnost objednat si jakékoliv jídlo není průměrná cena za jídlo závislá volbě na restaurace a je stejná.

Pro vegana navštěvujícího restaurace s masožravcem, který se podrobně dívá na ekonomiku cen na jídelním lístku, je však průměrná cena jídla vysoká. Pokud bychom chtěli snížit tuto průměrnou cenu, pak je nutné mít další informace, které by zohledňovaly, co si kdo dá a kolik by, v jaké restauraci by toto jídlo stálo.

Toto ve výsledku znamená, že zlepšení výkonnosti v řešení optimalizačních problémů je závislé na předchozí znalosti, jaké procedury jsou vhodné pro ten daný typ problému.

2.2 Stochastické algoritmy pro globální optimalizaci

Nemožnost nalézt optimalizační deterministický algoritmus, který by dokázal vyřešit všechny úlohy, vedla k tomu, že se při řešení problémů začalo využívat stochastických algoritmů. Ty sice nezaručují jisté vyřešení daného problému, nicméně nalezení dostatečného řešení v přijatelném čase je důležitým aspektem, který zatím normální deterministické optimalizační metody postrádají [14].

Stochastické algoritmy používají heuristiku při prohledávání prostoru řešení dané účelové funkce. Heuristické řešení se od deterministického liší tím, že heuristické řešení naproti deterministickému nezaručuje nalezené řešení. Řešení heuristikou tedy nemusí být nutně globálním řešením účelové funkce [14].

Algoritmy pro globální optimalizaci často čerpají ve svých strukturách z přírodních modelů, kde každý využívá určitý proces učení, díky němuž se sám algoritmus na základě jeho vnitřních pravidel orientuje v prostoru řešení. Značná část těchto algoritmů pracuje v prohledávaném prostoru řešení s více jedinci, pomocí kterých se hledá optimum. V tomto prostoru pak dochází k postupným krokům znamenajících pohyb či evoluci.

2.3 Evoluční algoritmus

Evoluční algoritmus je optimalizační a prohledávací metoda založená na principu evoluce a přirozeného výběru. Každý genetický algoritmus je složen z populace jedinců, kteří se vyvíjejí na základě pravidel evoluce do stavu, který minimalizuje hodnotu optimalizované účelové funkce. Tato metoda byla vyvinuta Hollandem [15] a dále rozvinuta jeho studentem Goldbergem [16]. Genetické algoritmy mají mnoho výhod. Dokáží například optimalizovat jak diskrétní, tak spojité proměnné a většinou oproti deterministickým metodám nejsou uvězněna v lokálních minimech.

Genetické algoritmy se dělí na dva základní typy [14]:

- Binární genetické algoritmy
- Kontinuální genetické algoritmy

Binární algoritmus bude v diplomové práci představen za účelem bližšího a názornějšího vysvětlení problematiky, ze které následně vychází spojitý GA, který bude představen také. Přes vzájemnou podobnost má však každý z algoritmů jisté pozitivní a negativní stránky, proto jsou v diplomové práci představeny a následně porovnány oba algoritmy.

Algoritmus, který bude mít v porovnání nejlepší výsledky, bude použit k tvorbě hybridního algoritmu.

2.3.1 Binární genetický algoritmus (bGA)

Genetický algoritmus popsán v této kapitole vychází z [15] a podobně jako jiné optimalizační metody začíná definicí proměnných, definicí účelové funkce a končí testem pro konvergenci. Stavové proměnné pro GA jsou definovány jako matice proměnných hodnot, které mají být optimalizovány. Pokud má chromozom N_{var} stavových proměnných které jsou dány p_1,p_2,p_3 atd., pak je chromozom napsán jako vektor o velikosti N_{var}.

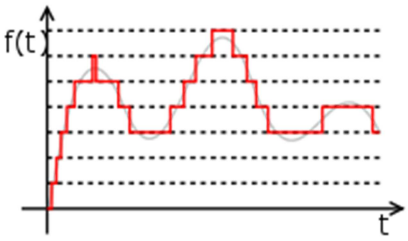
$$chromozom=[p_1, p_2, p_3, \dots, p_{Nvar}] \tag{39}$$

$$Hodnota\ účelové\ fce=f(chromozom)=f(p_1, p_2, p_3, \dots, p_{Nvar}) \tag{40}$$

Vzhledem k tomu, že je tento algoritmus binární a proměnné budou proto zaváděny v binárním kódu, musíme zahrnout určitý stupeň kvantování vyhovující řešení daného problému. Proměnná p_i má hodnotu reprezentovanou řetězcem bitů, který je N_{gene} dlouhý. Pokud má gen y bitů pak možností hodnot, kterých může nabývat je y^{Ngene}.

Kvantování

Vzhledem k tomu, že jsou stavové proměnné dány binárním kódováním, musíme zavést přístup, kterým se budou tyto hodnoty konvertovat z binárních do spojitých a naopak. Kvantování vychází z nepřekrývaného vzorkování do jednotlivých podmnožin, kde každé podmnožině je přiřazena jednotlivá diskrétní hodnota. Vzhledem k tomu, že dochází k rozdílu mezi hodnotou kvantované účelové funkce a hodnoty účelové funkce, je nutné zvolit kvantovací parametry s ohledem na možnost budoucí chyby vlivem kvantování [17]. Algoritmu pro převod binárních stavových proměnných na spojité proměnné je součástí Přílohy 2.



Obr. 8 Ilustrační obrázek kvantování spojité funkce

Přirozený výběr

Algoritmus pracuje na základě přirozeného výběru, a ten určuje přežití nejlepšího chromozomu. To stanovuje vyřadit po evaluaci účelové funkce chromozomy s horší hodnotou, než je ta, kterou má nejlepší jedinec.

Populace náhodně vytvořených jedinců, které ukazuje tab. 6, je seřazena podle sestupně podle hodnoty a nejlepší jedinci jsou vybráni pro pokračování v páření, zatímco ostatní jsou odstraněni z populace. Průběh seřazení a následného vyřazení ukazuje tab. 7.

Výběr probíhá na základě předem definovaného výběrového poměru. Tento přirozený výběr se děje v každé generaci, kterou v algoritmu zastupuje daná iterace. Jelikož je výběrový poměr exogenní k algoritmu, jeho hodnota může nabývat (0,1).

Tab. 6 Počáteční populace binárního genetického algoritmu

Pořadí	Chromozom	Hodnota ÚF
1	00101111000110	-12359
2	11100101100100	-11872
3	**00110010001100**	**-13477**
4	**00101111001000**	**-12363**
5	11001111111011	-11631
6	01000101111011	-12097
7	**11101100000001**	**-12588**
8	**11101100000001**	**-12588**

Malá hodnota výběrového poměru brzdí algoritmus ve vývoji, zatímco velká hodnota nechává k páření i horší jedince. V rámci konvence se volí výběrový poměr jako 50%, který byl také zvolen pro binární GA v diplomové práci.

Tab. 7 Selekce nejlepších jedinců v populaci

Pořadí	Pořadí minulé	Chromozom	Hodnota ÚF	
1	3	**00110010001100**	**-13477**	
2	7	**11101100000001**	**-12588**	
3	8	**11101100000001**	**-12588**	
4	4	**00101111001000**	**-12363**	
-	1	00101111000110	-12359	Vyřazen
-	6	01000101111011	-12097	Vyřazen
-	2	11100101100100	-11872	Vyřazen
-	5	11001111111011	-11631	Vyřazen

Selekce

Selekce je děj v algoritmu, kdy z jedinců, kteří přežili a mají nejmenší hodnoty účelových funkcí, jsou vybrány dva chromozomy pro páření a následné vytvoření nového potomka. Toto páření probíhá, dokud není zajištěno $N_{pop} - N_{keep}$ potomků, kteří nahradí vyřazené jedince. Tím je zajištěno, že nedochází k nárůstu nebo úbytku populace. Metod pro páření jedinců je několik:

1) **Páření na základě nejbližších dvou následujících jedinců**

Jedinci jsou vybráni do dvojice vždy jako sudý a lichý. Páření probíhá vždy první s druhým, třetí se čtvrtým a tak dále. Metoda dvou nejbližších jedinců tvoří nejzákladnější metodu výběru rodičů

2) **Páření na základě náhodného výběru dvou jedinců**

Tento přístup používá uniformní generátor náhodných čísel pro výběr chromozomu.

3) **Páření na základě váženého náhodného výběru (Roullet wheel)**

U tohoto typu páření má jedinec s nejlepší hodnotou účelové funkce největší pravděpodobnost selekce k páření a naopak jedinec s nejhorší hodnotou účelové funkce bude mít pravděpodobnost nejnižší. Náhodné číslo následně určí, kteří jedinci budou vybráni. Tento typ výběru se často také označuje jako výběr pomocí kola rulety.

Existují dvě základní techniky vážení. Vážení dle pořadí a vážení dle hodnoty účelové funkce.

- ***Vážení dle pořadí***

Tento přístup najde pravděpodobnost výběru pro jedince podle pořadí jedinců v dané generaci. Tato pravděpodobnost je dána vztahem:

$$P_n = \frac{N_{keep} - n + 1}{\sum_{n=1}^{N_{keep}} n} \tag{41}$$

Kumulativní pravděpodobnost je následně použita při výběru chromozomu. Náhodné číslo, které leží (0,1), je následně generováno. Od nejlepšího jedince se postupuje směrem k nejhoršímu a první chromozom, který má pravděpodobnost větší, než dané náhodné číslo, je vybrán do skupiny pro páření. Toto ukazuje tab. 8.

To však znamená, že jeden chromozom může být vybrán více než jednou. Výběr chromozomu více než jednou by znamenalo replikaci daného chromozomu do počtu 3 stejných chromozomů v další generaci.

Tento přístup není žádoucí při malé velikosti populace, jelikož pak vzrůstá pravděpodobnost uvíznutí v lokálním optimu, proto je při výběru chromozomů více žádoucí generovat druhý chromozom náhodně. Další možností pak je použít pro další chromozom stejnou techniku vážení dle pořadí, popřípadě jednoduše zvětšit velikost populace.

Tab. 8 Zobrazení vážení jedinců dle pořadí

n	Chromozom	P_n	$\sum_{i=1}^{n} P_i$
1	00110010001100	0.4	0.4
2	11101100000001	0.3	0.7
3	00101111001000	0.2	0.9
4	00101111000110	0.1	1.0

- ***Vážení dle hodnoty účelové funkce***

Pravděpodobnost selekce určitého chromozomu je dána funkční hodnotou účelové funkce v daném chromozomu a ne jejím pořadím. Normalizovaná hodnota účelové funkce je vypočtena pro každý chromozom tím, že dojde k odečtení nejnižší hodnoty účelové funkce vyřazených chromozomů $c_{N_{keep+1}}$ od hodnoty všech zbylých chromozomů ve skupině vybraných pro páření.

$$C_n = c_n - c_{N_{keep+1}} \tag{42}$$

Tím, že odečteme $c_{N_{keep+1}}$ zajistíme normalizaci jednotlivých hodnot jedinců v účelové funkci. P_n je pravděpodobnost, která je vypočtena jako:

$$P_n = \left| \frac{C_n}{\sum_m^{N_{keep}} C_m} \right| \tag{43}$$

Tento přístup zvýhodňuje lepší jedince, pokud je rozdíl jejich účelových funkcí podstatný a naopak rozdělení pravděpodobnosti je rovnoměrné, pokud se navzájem funkční hodnoty účelové funkce v jednotlivých chromozomech výrazně neliší.

Zde se obdobně jako případě vážení dle pořadí vyskytuje větší pravděpodobnost selekce stejného jedince pro páření a tím pádem také pravděpodobnost znásobení chromozomu do následující generace. Řešení tohoto problému je stejné, jako v předchozím případě. Pravděpodobnost se přepočítává každou generaci.

Tab. 9 Vážení jedinců dle hodnoty normalizované účelové funkce

n	Chromozom	$C_n = c_n - c_{N_{keep+1}}$	P_n	$\sum_{i=1}^{n} P_i$
1	00110010001100	-13477 + 12359 = -1118	0,53	0.53
2	11101100000001	-12588 + 12359 = -491	0,23	0.76
3	00101111001000	-12588 + 12359 = -491	0.23	0.99
4	00101111000110	-12363 + 12359 = -4	0.00(1)	1

4) Páření na základě turnajového výběru

Tento přístup detailně napodobuje soutěžní chování tím, že náhodně vybere malou podpopulaci chromozomů ze skupiny vybrané pro páření. Chromozom s nejnižší hodnotou účelové funkce v této podpopulaci se stane rodičem. Tento typ výběru rodičů je opakován, dokud není nalezeno optimální množství rodičů, kteří vytvoří zbývající množství potomků, nahrazujících vyřazené chromozomy.

V diplomové práci byla z výše uvedených metod vybrána selekce, která zvýhodňuje jedince dle jeho pořadí mezi jednotlivými chromozomy, tedy metoda založená na kole rulety. Výběr byl založen na četnosti použití této metody literatuře.

Páření

Páření je proces, kdy dochází k vytvoření jednoho nebo více potomků od rodičů, kteří byli vybrání určitým typem procesu selekce. Genetické uspořádání populace je limitováno stávajícími členy populace. Nejběžnější forma páření zahrnuje dva rodiče, z nichž jsou produkováni dva potomci.

Tab. 10 Princip páření v binárním algoritmu

3	*Matka*	*00000111001000*
2	**Otec**	**11101100000001**
5	Potomek$_1$	*00000***100000001**
6	Potomek$_2$	**11101***111001000*
3	*Matka*	*00101111001000*
4	**Otec**	**00101001000110**
7	Potomek$_1$	**0010100***1001000*
8	Potomek$_2$	*0010111***1000110**

Při páření je použit bod křížení (kinetochora), který je náhodně vybrán jako místo mezi prvními a posledními bity rodičů. Rodič$_1$ následně přenese část svého binárního kódu z levé části na potomka$_1$ a část binárního kódu z části pravé na potomka$_2$.

Rodič$_2$ pak stejným způsobem přenese levou část binárního kódu rozděleného bodem křížení na potomka$_2$ a pravou část binárního kódu na potomka$_1$. Rodiče v důsledku tohoto křížení vytvořili N_{pop}-N_{keep} potomků a populace chromozomů je nyní zpět na N_{pop}.

Mutace

Náhodná mutace mění některé bity v listu chromozomů. Mutace je dalším způsobem jak GA prozkoumává prostor řešení. Přispívá k představení vlastností, které nebyly součástí původní populace, a do jisté míry zabraňuje algoritmu konvergovat příliš rychle, bez prozkoumání většího prostoru řešení. Mutace mění v jednom bodě původní bit za bit opačný, jak ukazuje tab. 11.

Body, ve kterých se mutace vyskytne, jsou vybírány náhodně z matice N_{mut} *x* N_{bits}, kde N_{mut} jsou jedinci, u kterých lze provést mutaci. Zvětšováním počtu jedinců, u kterých dochází k mutaci, se, zvětšuje volnost algoritmu vůči prozkoumávání většího prostoru řešení. Počet jedinců, u kterých proběhne mutace, je dán mutačním poměrem, který může nabývat hodnot (0,1).

Mutace neprobíhá při poslední iteraci a mutace není povolená pro nejlepšího jedince. Počet mutací je dán mutačním poměrem, u kterého platí podobná pravidla jako pro výběrový poměr. Čím větší bude mutační poměr, tím bude algoritmus konvergovat pomaleji. Naopak čím menší mutační poměr, tím více bude řešení záviset na původní populaci jedinců.

Tab. 11 Mutace jednotlivých částí chromozomu jedince

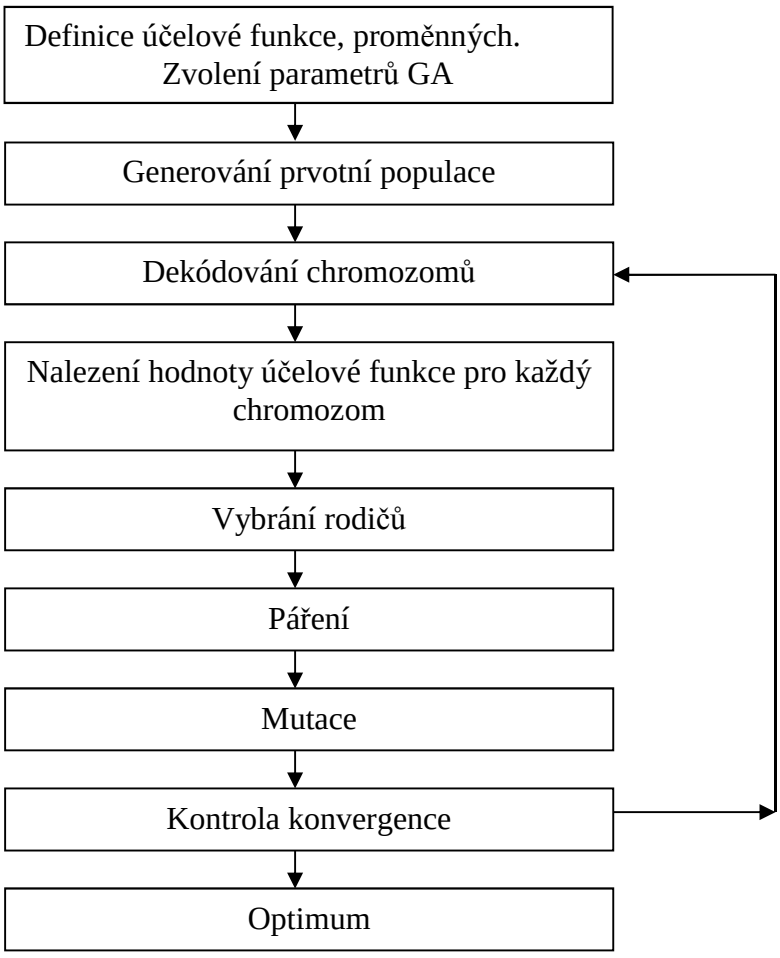
Před mutací	Po mutaci
11101100000001	11101100000000
00101000000001	00101100000001
11101111101000	11101111001000
00101111001000	10101111001000
00101111100110	00101111000110
00101111000110	00101111001110

Po mutaci populace jsou vyhodnoceny funkční hodnoty jednotlivých jedinců v daném prostoru řešení a následně jsou v další iteraci aplikovány metody přirozeného výběru, selekce, páření mutace a test pro konvergenci.

Konvergence binárního genetického algoritmu

Počet generací, které se vyvíjí, záleží obecně na podmínce, zda je dosaženo přijatelné řešení, nebo jestli je dosaženo maximálního počtu iterací. Evoluce generací způsobí, že po určitém počtu iterací všechny chromozomy, kromě chromozomů mutovaných, stejné. Konvergence je komplexní téma, které bude blíže představeno dále v diplomové práci ve stejnojmenné podkapitole. Celkový průběh algoritmu popisuje schéma na obr. 9. Lze vidět, že pro rychlost algoritmu bude žádané obejít dekódování proměnných v každé generaci.

Obr. 9 Schéma binárního genetického algoritmu



2.3.2 Spojitý genetický algoritmus (sGA)

Spojitý genetický algoritmus je vychází v této kapitole z [14]. Chromozom je u spojitého genetického algoritmu pole reálných čísel, které budou v evoluci optimalizovány, aniž by bylo potřeba jakékoliv kvantování. Pokud má chromozom N_{var} proměnných daných p_1, p_2 atd, pak je chromozom zapsán jako vektor N_{var} elementů způsobem:

$$chromozom=[p_1, p_2, p_3, \dots, p_{Nvar}] \tag{44}$$

Hodnotu účelové funkce každého jedince určíme jako funkční hodnotu účelové funkce v místě daného jedince. Jak již bylo uvedeno, odpadá mezikrok kvantování, kdy je nutné dekódování chromozomů.

$$Hodnota\ účelové\ fce=f(chromozom)=f(p_1, p_2, p_3, \dots, p_{Nvar}) \tag{45}$$

Na začátku je definována počáteční populace jedinců N_{pop}, která je reprezentována jednotlivými chromozomy. Vzhledem k tomu, že počáteční populace má N_{pop} chromozomů a je reprezentována N_{var} proměnnými, pak celá matice generovaných hodnot má velikost N_{var} x N_{pop}. Tab. 11 uvádí ilustrační příklad populace vytvořené pro dvě stavové proměnné a jejich účelové funkce.

Tab. 12 Počáteční populace spojitého genetického algoritmu

Pořadí	Chromozom	Hodnota ÚF
1	0.43869;0.10869	1.1211
2	0.38354;0.61845	1.0365
3	0.9595;0.67611	3.3679
4	***0.61511;0.78583***	***-1.4095***
5	***0.25748;0.30557***	***-2.5716***
6	***0.22531;0.12482***	***0.16834***
7	***0.11368;0.89967***	***-2.6063***
8	0.75975;0.72371	3.6966

Přírodní výběr a selekce

Stejně jako přírodní výběr v binárním genetickém algoritmu funguje přírodní výběr ve spojitém genetickém algoritmu. Dojde k ohodnocení jedinců a k jejich sestupnému seřazení od jedince s nejnižší funkční hodnotou účelové funkce.

Pomocí výběrového poměru je horší část populace vyřazena. Tento krok, stejně jako v binárním genetickém algoritmu, probíhá každou iteraci. Bez něj by nebyl možný vývoj populace.

Párování

Párování a párovací strategie jsou stejné jako u binární verze. Nejlepší jedinci jsou určitou párovací strategií vybrání pro následné tvoření N_{pop}-N_{keep} potomků. Zde byla v algoritmu použita metoda, která vážila jednotlivé chromozomy dle pořadí.

Tab. 13 Populace spojitého genetického algoritmu po selekci

Pořadí aktuální	Pořadí minulé	Chromozom	Hodnota ÚF
1	***7***	***0.11368;0.89967***	***-2.6063***
2	***5***	***0.25748;0.30557***	***-2.5716***
3	***4***	***0.61511;0.78583***	***-1.4095***
4	***6***	***0.22531;0.12482***	***0.16834***

Páření

Stejně jako v případě binárního algoritmu jsou dva rodičové vybráni pro páření a potomek bude kombinací informací rodičů. Metody páření využívají rovněž jednoho nebo více bodů křížení. Body křížení jsou náhodně vybraná místa v chromozomech rodičů, a informace nacházející se v daných oblastech je vyměněna. U přístupů páření spojitých genetických algoritmů bylo provedeno mnoho studií například [1]

Nejjednodušší metodou páření spojitého genetického algoritmu je vybrat jeden bod křížení, který rozdělí části chromozomu a prohodí rozdělené informace mezi rodiči pro vznik divergentních potomků.

Problém této metody je, že nedochází k vyvíjení populace, jelikož se nevytvářejí žádní jedinci, kteří by nebyli kombinací předchozí, náhodně iniciované populace. Tento problém vyřešilo prolnutí metody mutace a metody křížení pomocí kinetochor tím, že dojde ke zkřížení hodnot dvou rodičů do úplně nové hodnoty potomka. Jedinec, který vznikne je kombinací korespondujících hodnot potomků. [18]

$$p_{nové} = \beta p_{mn} + (1 - \beta) p_{dn} \tag{46}$$

Kde β je interval náhodné hodnoty (0,1); p_{mn} je *n*-tou proměnnou v matečném chromozomu; p_{dn}je *n*-tou proměnnou v otcově chromozomu;

Toto má za následek, že druhý potomek je komplementární k prvnímu záměnou β za (1- β). Pokud β = 1, pak p_{mn} je propagován celý a p_{dn} je vyřazen a naopak je tomu při β =0. Pokud je β = 0.5 pak jsou potomci průměrem rodičů.

Proces lineární kombinace je vyhotoven pro všechny proměnné na pravé nebo levé straně od bodu křížení. Proměnné mohou být kříženy vzhledem ke konstantnímu β po všechny generace, nebo může β nabývat jiných hodnot v každé generaci. Nejjednodušší metodu lineárního křížení popisují rovnice (47) a (48), kde dojde k vytvoření dvou potomků ze dvou rodičů:

$$p_{nové1=} 1.5 p_{mn} + 0.5\, p_{dn} \tag{47}$$

$$p_{nové2=} 1.5 p_{mn} - 0.5\, p_{dn} \tag{48}$$

Další možností je přístup heuristického překřížení, kde je koeficient β volen jako náhodná hodnota mezi 0 a 1 a proměnné potomků jsou následně definovány:

$$p_{nové=}\beta(p_{mn} - p_{dn}) + p_{mn} \quad (49)$$

V případě, že chceme napodobit výhody páření binárního genetického algoritmu, pak necháme rodiče [14]:

$$Rodič_1=[p_{m1},p_{m2},..,p_{mn}] \quad (50)$$

$$Rodič_2=[p_{d1},p_{d2},..,p_{dn}] \quad (51)$$

Bod křížení pak

$$a = roundup(random * N_{Var})+1 \quad (52)$$

$$p_{nové1} = p_{ma} - \beta(p_{ma} - p_{d\alpha}) \quad (53)$$

$$p_{nové2} = p_{da} - \beta(p_{ma} - p_{d\alpha}) \quad (54)$$

Kde β je náhodná hodnota (0,1) a posledním krokem je následně křížení zbytku chromozomů jako u binárního genetického algoritmu. Pokud je vybrán bod křížení jako druhý člen ve vektoru hodnot jedince, potomci nabývají následujícího tvaru [14]:

$$Potomek_1=[p_{m1},p_{m2}, p_{nové1},..p_{novéNvar-a}] \quad (55)$$

$$Potomek_2=[p_{d1},p_{d2}, p_{nové2},..p_{novéNvar-a}] \quad (56)$$

U potomků tedy dochází k vytvoření nových hodnot v chromozomu, které jsou dány kombinací hodnot rodičů od pozice vybrané bodem křížení.

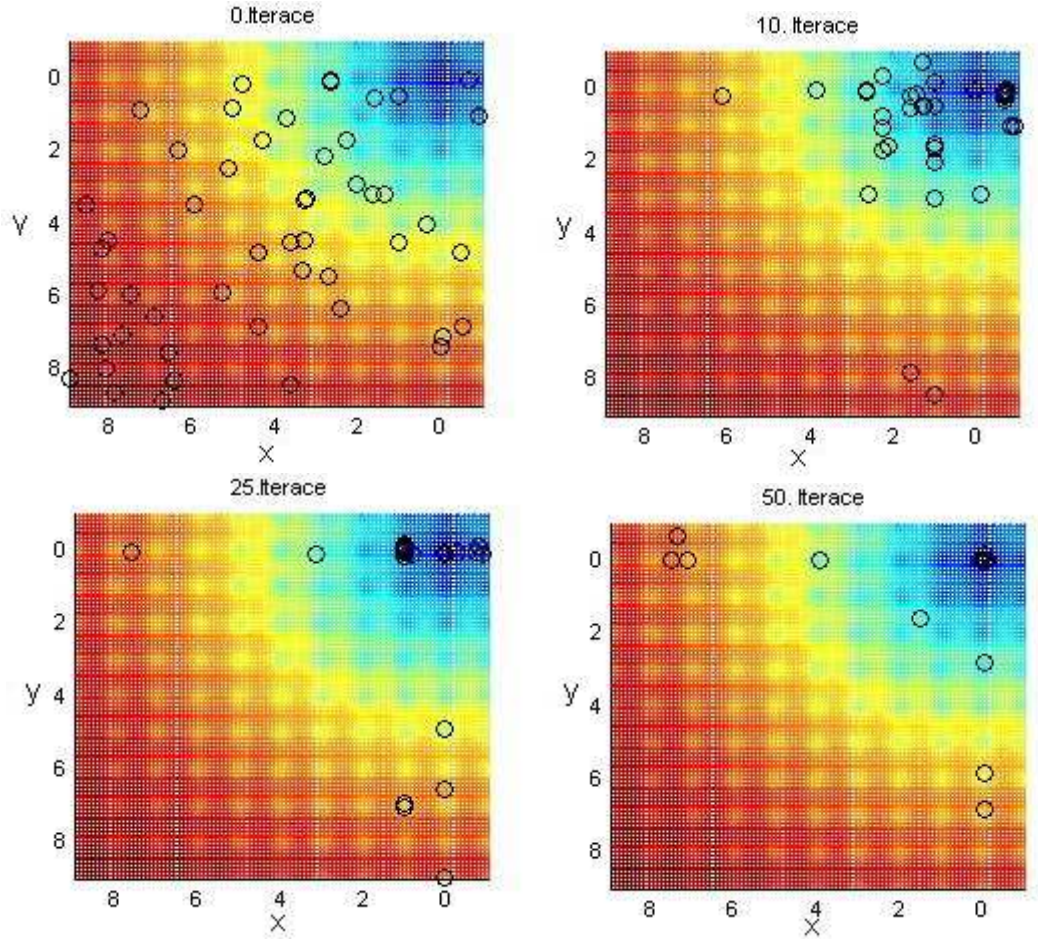
Mutace

Spojitý genetický algoritmu může stejně jako binární genetický algoritmus rychle konvergovat k lokálnímu optimu, které následně může nesprávně vyhodnotit jako globální optimum Proto je vhodné doplnit algoritmus o mutaci částí populace. Mutační poměr, který je přítomný v mutaci nám udává, u kolika procent populace dojde k mutaci. Není žádoucí, aby mutací prošel nejlepší jedinec.

Při mutaci jsou vygenerovány náhodné hodnoty pro určení řádku a sloupce, kde má mutace proběhnout. Mutace může probíhat záměnou za uniformně generované náhodné číslo, které leží v hranicích stejných jako počáteční populace. Další možností pro mutaci je následně generovat náhodně číslo v závislosti na normální distribuci.

Konvergence spojitého genetického algoritmu

Iterování a tedy vývoj populace probíhá, dokud se nesplní podmínky pro ukončení algoritmu, nebo pokud nedojde k maximálnímu počtu iterací. Platí zde stejný princip jako u binárního genetického algoritmu.



Obr. 10 Průběh evoluce evolučního algoritmu na oblasti přípustných řešení

Lepší znázornění konvergence binárního a spojitého genetického algoritmu uvádí obr. 10, kde je znázorněn vývoj populace v závislosti na dané iterace (generaci), ve které se populace jedinců nachází. Oblast přípustných řešení je na obrázku tvořena Ackleyho funkcí (viz dále v práci), která je pro větší názornost probíhající evoluce vychýlena, její globální minimum však stále leží v bodě [0,0].

Prvotní generace je náhodně rozložena v oblasti přípustných řešení, je dána 0. iterací. Po 10 iteracích už je patrný vývoj populace ve směru globálního minima. Také už dochází u několika jedinců ke shodné identifikaci nejnižší prozatímní hodnoty, nicméně algoritmus využívá mutace, pro prozkoumání další části oblasti přípustných řešení.

V 25. iteraci už většina populace leží v oblasti, kde se nachází globální minimum, nicméně jsou pořád patrné drobné rozdíly mezi jednotlivci. V 50. iteraci už algoritmus může vyhodnotit lokální minimum, kterého dosáhl jako globální minimum a tedy řešení daného minimalizačního problému.

Schéma spojitého genetického algoritmu je velmi podobné schématu binárního genetického algoritmu, které je zobrazeno na obr. 9 až na krok dekódování chromozomů, a proto nebude v diplomové práci opakováno.

2.4 Algoritmy modelující chování skupin

Algoritmy modelující chování skupin mají významné místo mezi stochastickými algoritmy hledající globální optimum. Tyto algoritmy modelují sociální chování například hejna nebo smečky a jsou často v literatuře označovány jako particle swarm inteligence metody, tedy metody inteligentního chování skupin [1].

Algoritmy jako optimalizace hejnem částic (Particle swarm optimisation - PSO) jsou podobné evolučním algoritmům ve smyslu interakce více kandidátů řešení. Rozdíl je v tom, že u evolučních algoritmů máme populaci a hledáme pomocí křížení a mutace jejich nejsilnější jedince v rámci vývoje dané populace. U algoritmů modelující chování skupin to je skupina, kde se však její pohyb po prostoru řešení provádí za pomoci pravidel skupinového chování [19].

Mezi nejpopulárnější algoritmy v této kategorii je optimalizace hejnem částic (PSO) a optimalizace mravenčí kolonií (Ant colony optimisation - ACO). Pro účely hledání globálního řešení CGAM problému byl zvolen PSO algoritmus, který bude dále představen.

2.4.1 Optimalizace hejnem částic (PSO)

Tato technika byla vytvořena Kenedym [19] v polovině devadesátých let a podle ní je také zpracována tato část diplomové práce. Algoritmus byl inspirován chováním hejn ryb a ptáků, kde každý jedinec se chová jak individuálně, tak je i zároveň ovlivněn chováním hejna a jeho ostatními členy. Hejno je tvořeno N_{pop} jedinci, které hledá globální optimum v prostoru řešení v etapách.

Na počátku je hejno N_{pop} částic, které jsou náhodně rozmístěny po prostoru řešení. Každá částice si pamatuje svou individuální optimální polohu a hejno jako celek si pamatuje optimální polohu nejlepšího řešení, kterého dosáhlo celé hejno. Částicím je přiřazována poloha a rychlost v jednotlivých etapách prozkoumávání. Počáteční poloha a rychlost se volí náhodně. Algoritmus následně iteruje základní rovnici po určitý počet předem daných iterací, nebo dokud se nesplní ukončující podmínka.

Pohyb jedince

Jedinec mění pozici v prostoru svou rychlostí, která se mění v závislosti na etapách, ve kterých se nachází [19].

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{57}$$

Pozici *i*-tého jedince na začátku etapy určíme jako součet jeho současné pozice a dráhy, kterou jedinec v nynější etapě urazí. Etapa trvá jednotkový čas. Rychlost jedince v aktuální etapě vychází z jeho minulého pohybu a jeho minulé rychlosti, ale také z minulého pohybu hejna. Hejno jako takové má informace o prohledávaném prostoru a ovlivňuje chování jednotlivce. Rychlost daného jedince v závislosti na hejnu lze vyjádřit [19]:

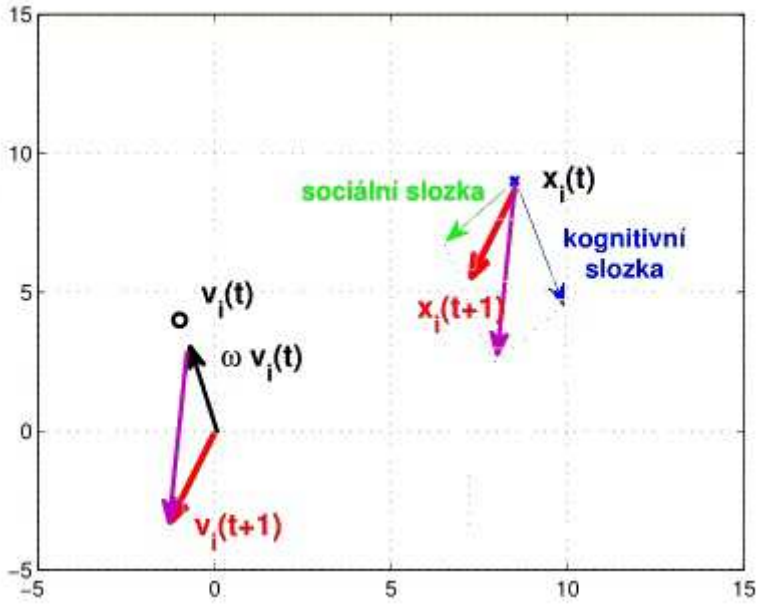
$$\boldsymbol{v}_i(t+1) = \omega \boldsymbol{v}_i(t) + \varphi_1 \boldsymbol{u}_1 \otimes (\boldsymbol{p}_i - \boldsymbol{x}_i) + \varphi_2 \boldsymbol{u}_2 \otimes (\boldsymbol{g} - \boldsymbol{x}_i) \tag{58}$$

Koeficienty ω a φ_1a φ_2 jsou řídící vstupní parametry a vektory $\boldsymbol{u}_1$ a $\boldsymbol{u}_2$ jsou vektory dimenze *d*, jejíž prvky jsou nezávislé náhodné hodnoty rovnoměrně rozdělené na intervalu (0,1). Počet stavových proměnných jednotlivých funkcí je dáno dimenzí problému *d*.

Rychlost jedince je v rovnici (57) tvořena třemi složkami, které vytvářejí navzájem se ovlivňující vektory. Rychlostí daného jedince v minulé etapě $\boldsymbol{v}_i(t)$, což je setrvačná složka vyjadřující směr, kterým se pohybuje jedinec. Tato rychlost je násobená váhou setrvačnosti omega, která se volí v intervalu (0,1).

Kognitivní složku tvoří v rovnici (58) druhý člen $\varphi_1 \boldsymbol{u}_1 \otimes (\boldsymbol{p}_i - \boldsymbol{x}_i)$. Ten je určován minulostí jedince. Zde p_i je poloha bodu s nejmenší funkční hodnotou, kterou daná jedinec našel od začátku prohledávání.

Sociální složku tvoří v rovnici (58) třetí člen $\varphi_2 \boldsymbol{u}_2 \otimes (\boldsymbol{g} - \boldsymbol{x}_i)$. Ten je určován minulostí celého hejna kde bod g odráží polohu bodu s nejmenší funkční hodnotou, která byla od začátku prohledávání nalezena celým hejnem.



Obr. 11 Skládání vektorů určujících pohyb částic [20]

Obr. 11 zobrazuje jaký geometrický význam má skládání vektorů a jak je celkový směr částice ovlivněn předchozím pohybem hejna a předchozími minimy a maximy.

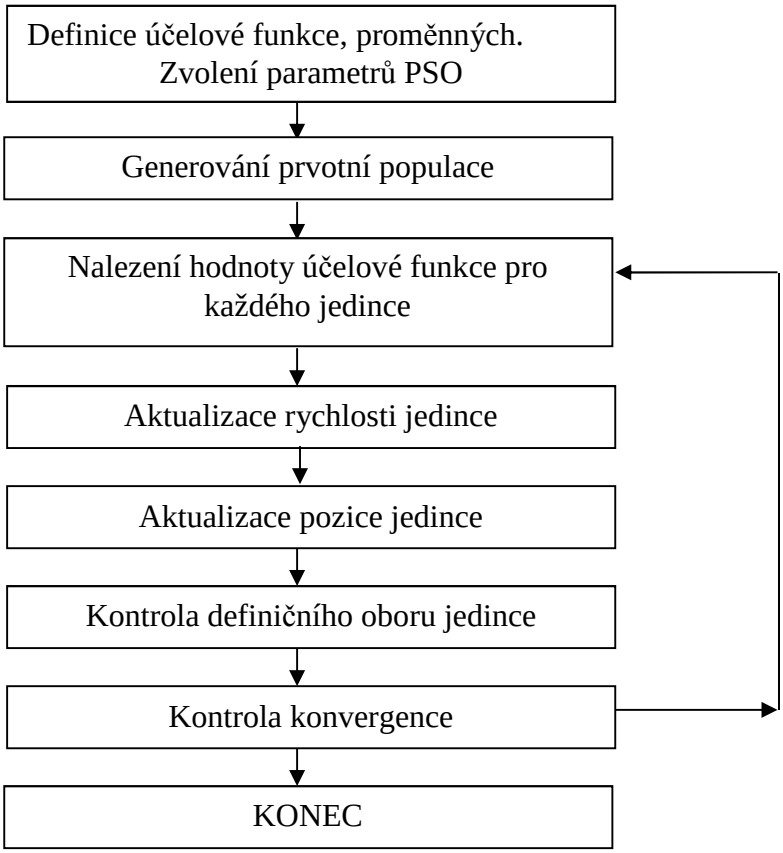
Částice se po prostoru pohybují volně, a díky tomu se mohou dostat i mimo definiční obor, který chceme vůči účelové funkci řešit. Je proto nutné kontrolovat jejich hodnoty a v případě, kdy by došlo k překročení definičního oboru změnit polohu částice.

Změna polohy částice může být provedena tzv. odrazem nebo absorpcí na hranici definičního oboru. Odraz je proveden překlopením souřadnic jedince, který se dostal mimo prohledávaný prostor dovnitř prohledávaného prostoru kolem příslušné hrany daného prostoru. Absorpce naopak jedinci přiřazuje hodnoty, které by měl na hranici definičního oboru [20].

Algoritmus optimalizace hejnem byl zpracováván s kontrolou definičního oboru s absorpcí. Tato metoda je jednodušší a v praxi stejně tak účinná, jako metoda odrazu částice zpět do prostoru řešení.

Průběh jednotlivých kroků algoritmu hejna částic shrnuje schéma na obr. 12.

Obr. 12 Schéma algoritmu optimalizace hejnem částic

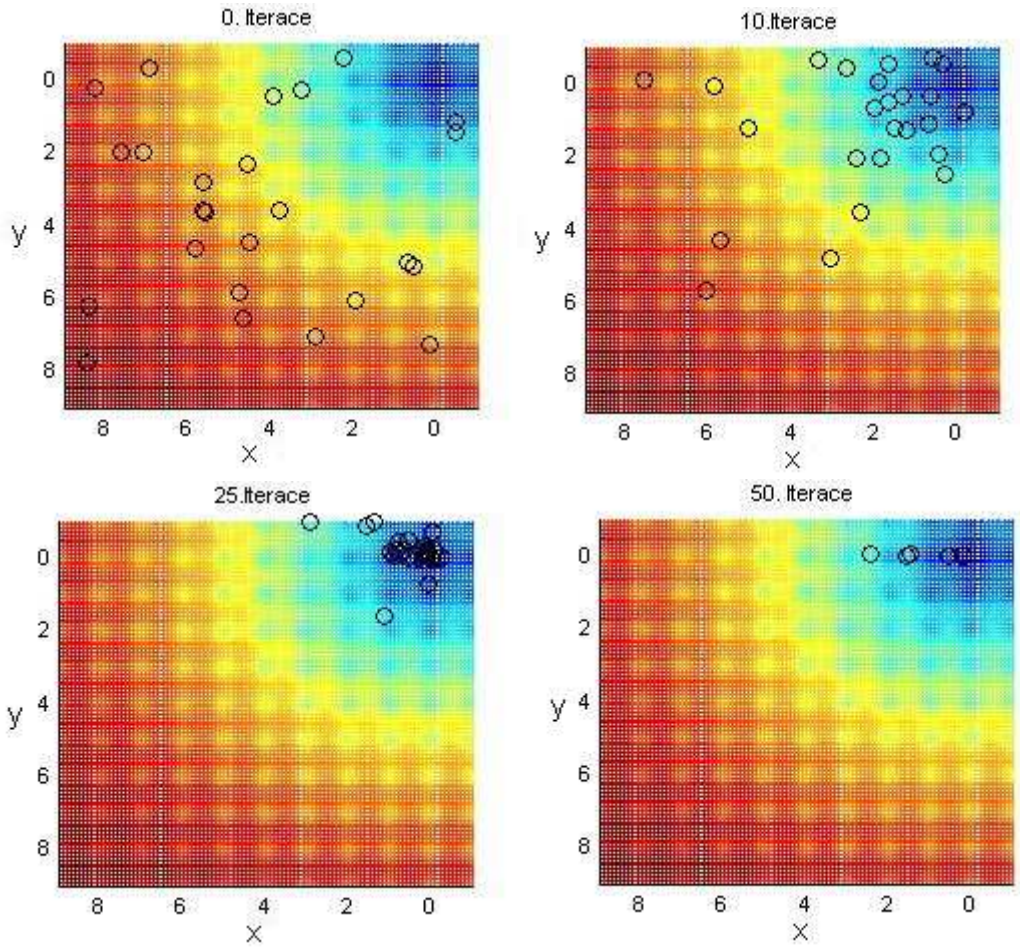


Podobně jako u evolučních algoritmů pro lepší znázornění konvergence algoritmu hejna uvádí pohyb populace obr. 13. Oblast přípustných řešení je na obrázku tvořena Ackleyho funkcí (viz dále v práci), která je pro větší názornost probíhajícího pohybu vychýlena, její globální minimum však stále leží v bodě [0,0].

Prvotní generace je náhodně rozložena v oblasti přípustných řešení, je dána 0. iterací. Na části obr. 13 znázorňující 10. iteraci lze pozorovat pohyb vzdálenějších částic hejna k nejoptimálnější hodnotě, kterou hejno dosáhlo.

Rozdíl oproti evolučním algoritmům je patrný až od 25. iterace, kdy je patrné, že algoritmus netvoří žádné jedince mutací a tedy téměř všechny částice se nachází v oblasti, kde se reálně nachází globální minimum Ackleyho funkce. Lze také vidět, že někteří jedinci byli absorbováni hranici oblasti přípustných řešení.

V 50. iteraci na obr. 13 už algoritmus může vyhodnotit minimum, které na oblasti přípustných řešení nalezl jako globální minimum a tedy řešení daného problému



Obr. 13 Průběh pohybu algoritmu hejna částic na oblasti přípustných řešení

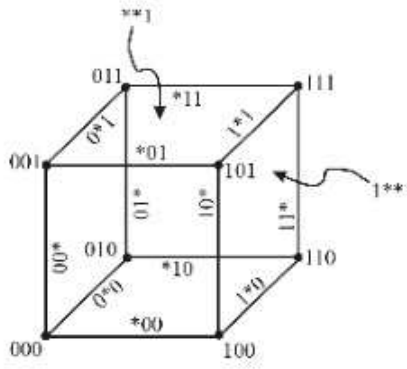
2.5 Konvergence

Konvergence genetických algoritmů a stochastických algoritmů vůbec je důležité téma, kterému se mnozí věnovali za pomocí Markovových řetězců. Ve studiích je všeobecně spoléháno na velký počet jedinců v populaci spolu s malým mutačním poměrem. Studie se soustředily na kritickou velikost populace a na počet potřebných iteračních kroků pro dosažení určitého řešení v daném intervalu spolehlivosti. Dle literatury je nejvíce používaným důkazem konvergence pro genetické a stochastické algoritmy [14].

2.5.1 Teorém schématu

Schématem je dle Goldberga [16] řetězec znaků, který se skládá z binárních čísel 0 a 1 a doplňkového znaku *, nahrazujícího 0,1 v náhodném výběru. To znamená, že řetězec 11**00 může reprezentovat čtyři další řetězce, které vzniknou z dosazení všech možných kombinací.

Na obr. 14 lze vidět krychli, ve které tři binární číslice reprezentují roh, dvě binární číslice a doplňkový charakter * reprezentují hranu a jeden binární charakter a dva doplňkové charaktery reprezentují plochu. Pokud pak 1** reprezentuje minimální hodnotu účelové funkce, pak je tato plocha oblastí zájmu pro genetický algoritmus.



Obr. 14 Goldbergova krychle [16]

Schéma, kde se vyskytují lepší, než průměrné výsledky pro účelovou funkci se vyskytuje exponenciálně více frekventovaně v každé další následující generaci. Schéma, které má výsledky účelové funkce horší než průměrné se vyskytuje méně frekventovaně v následující generaci.

Hlavní myšlenkou je, že nejlepší jedinci přežijí do dalších generací. Potom při následování nejlepšího schématu genetický algoritmus konverguje k jednomu jedinci a tudíž chromozomu z populace. Tento jedinec však nemusí být globálním minimem.

V praxi se uvádí mnoho přístupů, které jsou odvozeny od tohoto teorému, nicméně neexistuje zde jednotná odpověď s ohledem na fuzzy aspekty genetických algoritmů. Pro základní zjištění, zda algoritmus konvergoval a zda konvergoval ke globálnímu minimu. Lze použít několik možností [14]:

1) Možnost správné odpovědi tj. zastavit algoritmus, pokud je výsledný nejlepší chromozom přijatelným řešením problému. V praxi však ale není možné mít optimální hodnoty komplexního kogeneračního systému před jeho řešením.
2) Možnost žádného zlepšení. Tato možnost vyplývá z logiky genetického algoritmu, kdy dochází k multiplikaci nejlepšího řešení. Pokud algoritmus replikuje nejlepší chromozom po *n* iterací, pak lze algoritmus zastavit. Zde je důležité mít velkou populaci jedinců a nechat prostor algoritmu a mutačnímu parametru možnost najít i lepší řešení. Pokud bude v algoritmu malá populace, pak dojde ke konvergenci velice rychle a bude zde velká pravděpodobnost špatného řešení.
3) Možnost použití statistiky. Pokud průměrná hodnota účelové funkce dosáhne určité hodnoty, popřípadě je směrodatná odchylka řešení v rámci generace na určité předem definované hodnotě, pak se v populaci nevytváří žádná nová řešení a lze algoritmus zastavit.
4) Možnost iteračního odpočítávání. Dokud algoritmus nedosáhne přiděleného počtu iterací, tak pokračuje v evoluci jedinců v populaci.
5) Za pomoci lokální optimalizace.
6) Pomocí různé kombinace přechozího.

Obecně lze říci, že pokud algoritmus nekonverguje do dobrého řešení, lze změnit základní důležité parametry jako velikost populace a mutační poměr. Velikost populace má zásadní vliv na výkonnost algoritmu, ale zvyšuje časovou náročnost výpočtu. Genetický algoritmus tedy nemusí nutně najít globální minimum i přesto, že jsme jej několikrát aplikovali na určitý problém. V praxi však může být i toto řešení dostačující [1].

Jako kritéria konvergence ve vytvářených algoritmech bylo v diplomové práci zvoleno kritérium žádného zlepšení, kdy byl algoritmus nastaven na hodnotu maximálně patnácti opakování nejnižší hodnoty účelové funkce.

Pokud tedy algoritmus v patnácti generacích nenašel lepší řešení, než jakým je řešení uvedené v předchozí generaci, ukončil výpočet. Toto bylo doplněno kritériem maximálního počtu iterací, a u každého algoritmu byl maximální počet iterací nastaven na hodnotu 200.

Tyto dvě kritéria byla logicky sečtena, a zastavení algoritmu bylo způsobeno kritériem, u kterého se pravda objevila nejdříve.

3 Porovnávání a ověřování algoritmů

Porovnávání dvou a více algoritmů se provádí na základě stejných podmínek a na stejných testovacích funkcích, ve stejném testovacím prostoru řešení a při stejných podmínkách ukončení prohledávání prostoru řešení.

Základními veličinami pro porovnávání algoritmů je časová náročnost prohledávání a spolehlivost nalezení globálního minima. Časová náročnost se ohodnocuje počtem iteračních vyhodnocení účelové funkce. Spolehlivost nalezení globálního minima pak tím, zdali algoritmus konvergoval do předem známého intervalu hodnot, který byl vyhodnocen exogenně jako dostačující řešení.

Počet iterací, které jsou potřeba k vyhodnocení určité účelové funkce, je označen *n* (počet iterací funkce). Toto označení lze využít u účelových funkcí, kde je známo řešení a je důležité vědět, za jak rychle se algoritmus přiblíží ke globálnímu minimu. Vyhodnocení probíhá zvolením hodnoty, pod kterou když jedinec populace klesne, pak je dané řešení dostatečně blízko řešení globálnímu.

Podmínka úspěšného ukončení je v algoritmech dána jako:

$$fmin_{i-x} = fmin_i \tag{59}$$

Kde fmin_i tvoří nejmenší funkční hodnota účelové funkce v aktuální *i*-té populaci a $fmin_{i-x}$ tvoří nejmenší funkční hodnota v minulé populaci, která se odehrála *x* generací nazpět. Prohledávání prostoru řešení pak končí, když se tyto dvě hodnoty rovnají.

Druhou podmínkou ukončení, která zabraňuje algoritmu dostat se do nekonečného cyklu je podmínka maximálního počtu iterací. Parametr *maxit* udává maximální povolený počet iterací účelové funkce a tuto podmínku lze formulovat jako :

$$fmin_{i-x} = fmin_i \vee pif \geq maxit \tag{60}$$

Hledání tedy pokračuje tak dlouho, dokud se nesplní alespoň jedna z podmínek.

Možnosti ukončení u algoritmů lze shrnout do:

1) Ukončení, ke kterému dojde jako $fmin_{i-x} = fmin_i \vee pif \geq maxit$. Zde algoritmus splnil podmínku ukončení a současně se přiblížil dostatečnému řešení před dosažením maximálního počtu iterací.
2) Pomalá konvergence a prohledávání bylo ukončeno maximálním povoleným počtem iterací.
3) Předčasně konverguje, ale nebylo nalezeno globální minimum. Algoritmus prohledávání skončil v lokálním minimu nebo se body populace přiblížily k sobě natolik, že jsou jejich funkční hodnoty velmi blízké
4) Selhání, kdy byl algoritmus ukončen po maximálním dovoleném počtu iteraci, aniž by bylo nalezeno globální minimum v prostoru řešení.

Po prohledávacích algoritmech globální optimalizace chceme, aby uměly nacházet řešení dostatečně blízké globálnímu minimu co nejrychleji, co nejspolehlivěji, a aby ukončení proběhlo ve vhodném stádiu prohledávání. Jediné zcela úspěšné prohledávání pak je typu 1, které považujeme za korektní ukončení. Vzhledem k náročnosti prohledávání určitých prostorů řešení však lze přijmout jako úspěšně řešení typ 2 [20].

Spolehlivost *R* nalezení globálního minima lze charakterizovat dle rovnice (61) [20]:

$$R = \frac{n_1}{n_2} \tag{61}$$

Kde n_1 je četnost ukončení prvního typu; n_2 četnost ukončení druhého typu.

V praxi se však může stát, že algoritmus často konverguje do lokálního minima, které je však také vhodným řešením problému. Proto lze vztah pro spolehlivost *R*, která je uvedena rovnicí (61), zmírnit na vztah (62), který bere řešení prvního a druhého typu jako úspěšné řešení a řešení třetího a čtvrtého typu bere, jako selhání, kdy algoritmus nebyl schopen nalézt řešení daného problému a to ať z důvodu předčasné konvergence do lokálního minima nebo z důvodu celkového selhání algoritmu typu 4. Celková spolehlivost *R* po zmírnění nabývá vztahu:

$$R = \frac{n_1 + n_2}{n_1 + n_2 + n_3 + n_4} \tag{62}$$

Kde n_1 je četnost ukončení prvního typu; n_2 je četnost ukončení druhého typu; n_3 je četnost ukončení třetího typu; n_4 je četnost ukončení čtvrtého typu.

Výsledky testování na genetických algoritmech a na algoritmu optimalizace hejnem zahrnují určení parametrů testovaných algoritmů, jako například použitá velikost populace, specifikace testovacích funkcí, počet opakování experimentů, tabulku průměrných hodnot charakterizující časovou náročnost a charakteristiky variability a spolehlivosti pro jednotlivé algoritmy.

U algoritmů se sledují také veličiny, které zaznamenávají průběh vyhledávacího procesu. Graf konvergence následně zahrnuje vývoj nejlepších hodnot v dané populaci vzhledem k počtu iterací. Tímto lze snadno vizuálně porovnat jednotlivé algoritmy vzhledem k rychlosti jejich konvergence v různých iteračních fázích algoritmů.

K rychlému porovnávání účinnosti algoritmů je v diplomové práci navržena další veličina integrující časovou náročnost i spolehlivost do jediného kritéria. Jedná se o tzv. *Q*-míru definovanou vztahem (64).

$$n_p = \frac{n_1 + n_2 + n_3 + n_4}{4} \tag{63}$$

$$Q = \frac{n_p}{R} \tag{64}$$

kde n_p je průměrný počet vyhodnocení funkce v opakovaných běžích algoritmu při řešení úlohy; R je spolehlivost definovaná vztahem (62).

Nejúčinnější algoritmus mezi porovnávanými je ten s minimální hodnotou *Q*, jelikož zde platí nepřímá úměra mezi průměrným počtem iterací a spolehlivostí daného algoritmu.

Kritéria integrující n_p a *R* jsou vhodná pro rychlé přehledné porovnávání celkové účinnosti algoritmů, nelze z nich však dělat úsudky o variabilitě časové náročnosti algoritmů, ani jednoznačně posoudit jejich spolehlivost.

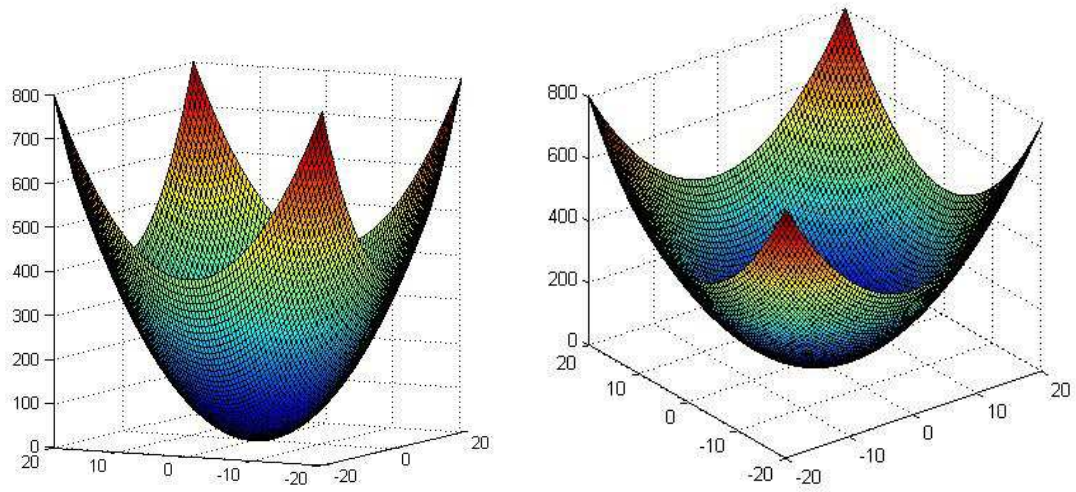
Pro porovnání algoritmů jsou v diplomové práci zohledněny i ostatní výše uvedené parametry jednotlivě, například n_p, popřípadě směrodatná odchylka řešení, která ve více běžích číselně udá přesnost nalezeného minima algoritmu vůči reálnému globálnímu algoritmu.

3.1 Testovací funkce

K testování stochastických algoritmů se využívají funkce, u nichž je předem známo řešení a toto řešení tvoří globální minimum. Testovací funkce jsou unimodální a multimodální. Unimodální funkce mají jedno lokální minimum, které je tedy i globálním minimem. Multimodální funkce jsou funkce, které mají více lokálních minim a jedno globální minimum. Počet stavových proměnných jednotlivých funkcí je dáno dimenzí problému *d*.

Zde si uvedeme několik testovacích funkcí, které pomohou zkontrolovat funkčnost algoritmu, a na těchto funkcích bude možné zjistit výkonnost jednotlivých algoritmů.

3.1.1 De Jongova funkce



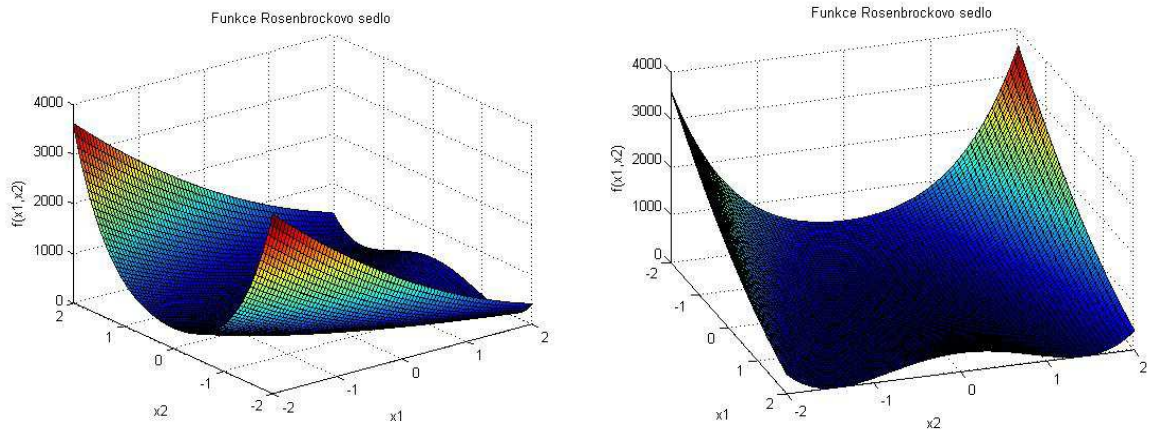
Obr. 15 Ilustrace De Jongovy funkce v prostoru

De Jongova funkce [21] má jednoduchý tvar paraboloidu, který je dán rovnicí (65):

$$f(x) = \sum_{i=1}^{d} x^2 \tag{65}$$

Je to konvexní separabilní funkce a její globální minimum je v bodě x* =[0, 0]. Tato funkce by neměla tvořit pro algoritmy výrazný problém, nicméně lze dle literatury předpokládat, že jednodušší metody optimalizace zde budou konvergovat rychleji, než složitější metody.

3.1.2 Rosenbrockovo sedlo



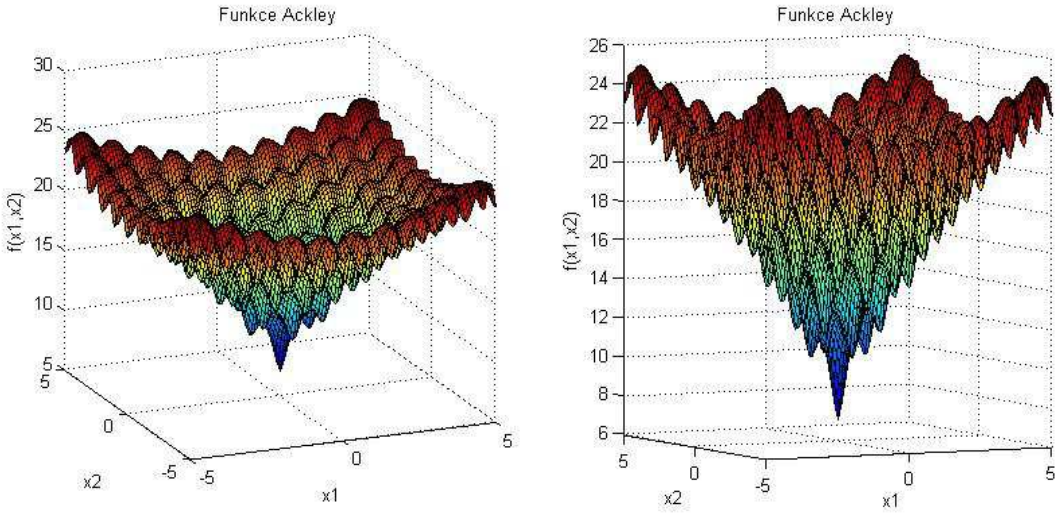
Obr. 16 Ilustrace funkce Rosenbrockova sedla

Rosenbrockovo sedlo [21] je funkce se dvěma minimy, které tvoří také globální minima. Jejich souřadnice jsou v bodech x_1* =[1, 1] a x_2* =[1, -1]

$$f(x) = \sum_{i=1}^{d-1} 100(x_i{}^2 - x_{i+1})^2 + (1 - x_i)^2 \tag{66}$$

Nalezení globálního minima je pro stochastické algoritmy zdlouhavé, případně tyto algoritmy předčasně ukončují svůj průběh, jelikož globální minimum leží v údolí s velmi nízkým spádem.

3.1.3 Ackleyho funkce



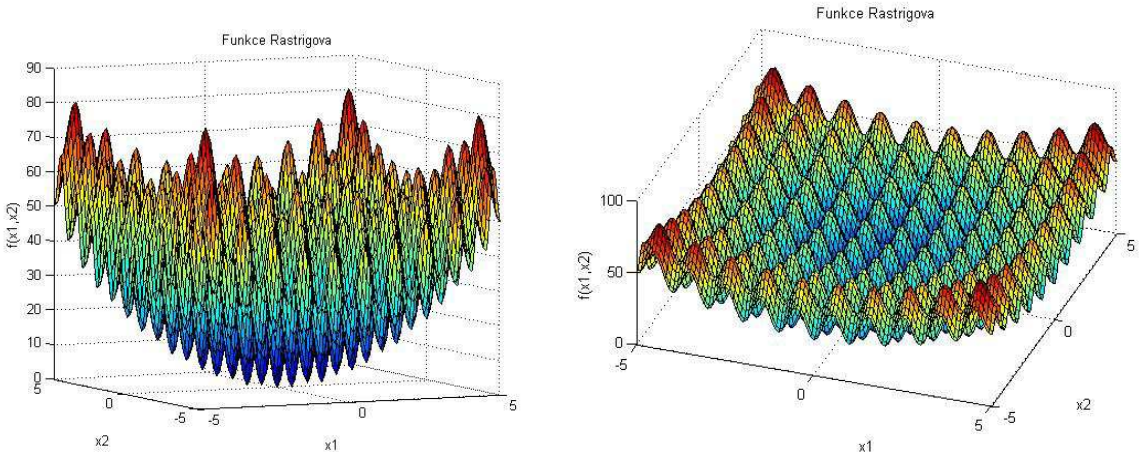
Obr. 17 Ilustrace Ackleho funkce v prostoru

Ackleyho funkce je dle [21] multimodální funkce s mnoha lokálními minimy, které mezi sebou však mají výraznější spád účelové funkce. Globální minimum funkce je x* =[0, 0].

$$f(x) = -20\exp\left(-0.02\sqrt{\frac{1}{d}\sum_{i=1}^{d} {x_i}^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d} cos2\pi\, x_i\right) + 20 + \exp(1) \tag{67}$$

Ackleyho funkce je považována za středně obtížnou funkci pro výpočetní algoritmy.

3.1.4 Rastrigova funkce



Obr. 18 Ilustrace Rastrigovy funkce v prostoru

Rastrigova funkce [21] je multimodální, separabilní funkce, která je dána rovnicí (68).

$$f(x) = 10d + \sum_{i=1}^{d} ({x_i}^2 - 10\cos(2\pi x_i)) \tag{68}$$

Globální minimum funkce je v bodě x* =[0, 0]. Tato funkce je považována za obtížnou úlohu optimalizace. Rozdíl v obtížnosti řešení mezi Ackleyho funkcí a Rastrigově funkci je v malém spádu lokálních minim, která obklopují globální minimum.

3.2 Porovnávání algoritmů

Experimentální porovnání jednotlivých algoritmů na představených testovacích funkcích probíhalo ve vícedimenzionálním problému, kdy byly optimalizovány dvě stavové proměnné. Pro každou úlohu bylo provedeno 100 opakování a ve všech úlohách byla stejná podmínka ukončení, která sestávala z kritéria $fmin_{i-x} = fmin_i \vee pif \geq maxit$, kde x bylo zvoleno jako hodnota 15.

Sledována byla spolehlivost algoritmů v nacházení řešení, časová náročnost algoritmu pro dosažení podmínky pro potřebný počet iterací pro ukončení, a přesnost řešení.

Testovací funkce plnily dvě základní úlohy. První úlohou byla kontrola napsaných algoritmů, zda dokážou najít správná řešení. Druhou úlohou bylo následné porovnávání jednotlivých aspektů algoritmů.

Binární genetický algoritmus a spojitý genetický algoritmus byly nastaveny v hlavních parametrech stejně (velikost populace, mutační poměr a metody výběru jedinců pro páření). Pro všechny algoritmy platilo jednotné nastavení ukončení algoritmu $fmin_{i-15} = fmin_i$.

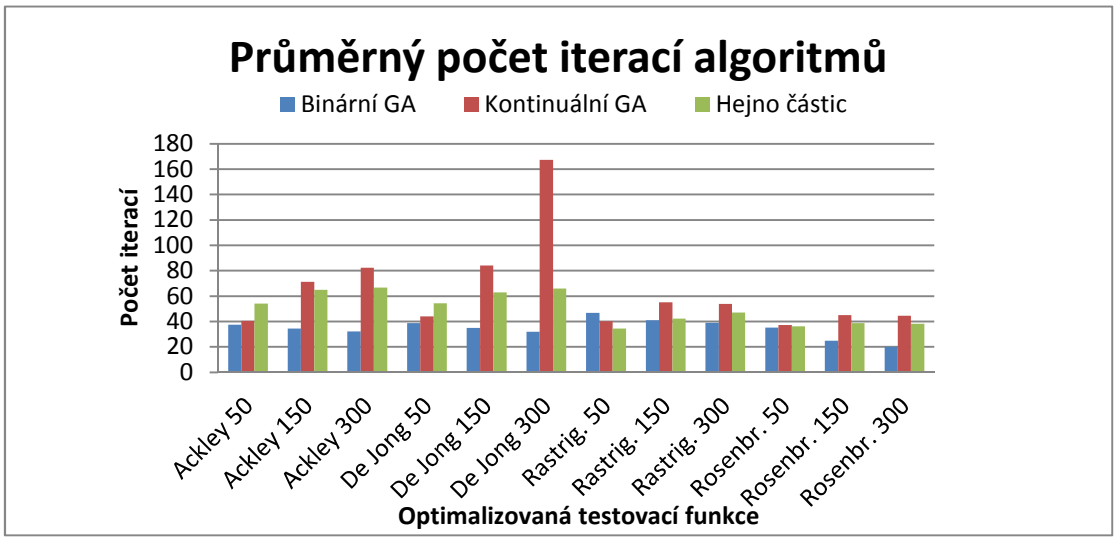
Porovnávání algoritmů bylo provedeno na základě měřítek velikosti populace, průměrného počtu úspěšných řešení, pravděpodobnosti nalezení úspěšného řešení a směrodatné odchylky výsledného řešení účelové funkce, jak je uvedeno v tab. 14.

Byly zvoleny tři základní velikosti populace, pro které se daný algoritmus testoval. Velikost 50, 150 a 300 jedinců. Úspěšné řešení je dáno jako hodnota účelové funkce s dodatečnou nadhodnotou, která byla vybrána jako 0,1. Jednotlivé funkce měly intervaly úspěšného řešení:

- Ackleyho funkce: $f(x^*) \in \langle 7{,}125, 7{,}225\rangle$;
- De Jongovy funkce: $f(x^*) \in \langle 0, 0{,}1\rangle$;
- Rastrigova funkce: $f(x^*) \in \langle 0, 0{,}1\rangle$;
- Rosenbrockovo sedlo: $f(x^*) \in \langle 0, 0{,}1\rangle$;

Na základě počtu úspěšných řešení se následně počítala spolehlivost nalezení úspěšného řešení jako podíl počtu úspěšných pokusů, k celkovému počtu měření. Celkový počet měření pro každý algoritmus byl zvolen 100.

Nejrychlejší z algoritmů se ukázal, jak ukazuje obr. 19, binární genetický algoritmus, u kterého byl průměrný počet iterací do konvergence 34,72. Druhým algoritmem byla z hlediska konvergence optimalizace hejnem částic, kdy byl průměrný počet iterací do konvergence 50,49. Nejpomalejším algoritmem byl spojitý genetický algoritmus, kde byl průměrný počet iterací do konvergence 63,72.



Obr. 19 Průměrný počet iterací jednotlivých algoritmů na testovacích funkcích (číslo za názvem funkce označuje velikost populace)

Tab. 14 Výsledky jednotlivých algoritmů na testovacích funkcích

Binární GA

Funkce	Ackley	Ackley	Ackley	De Jong	De Jong	De Jong
Vel. Populace	50	150	300	50	150	300
Prům. počet it.	37,55	34,36	32,13	38,73	34,9	31,99
Míra Q	37,55	34,36	32,13	38,73	34,9	31,99
Spolehlivost R	100%	100%	100%	100%	100%	100%
STD účelové funkce	0,013698	0,00588	0,003263	0,000232	2,7E-05	1,91E-05

Spojitý GA

Vel. Populace	50	150	300	50	150	300
Prům. počet it.	40,4	71,21	82,43	44,04	84,14	167,35
Míra Q	50,5	71,21	82,43	44,04	84,14	170,76
Spolehlivost R	80%	100%	100%	100%	100%	96%
STD účelové funkce	0,110841	0,001896	3,75E-07	0,002075	1,28E-05	0,052778

Hejno částic

Vel. Populace	50	150	300	50	150	300
Prům. počet it.	54,07	65,01	66,75	54,28	62,85	65,97
Míra Q	245,77	79,28	70,26	54,28	62,85	65,97
Spolehlivost R	22%	82%	95%	100%	100%	100%
STD účelové funkce	0,36421	0,045331	0,029256	0,006977	0,000974	0,000318

Binární GA

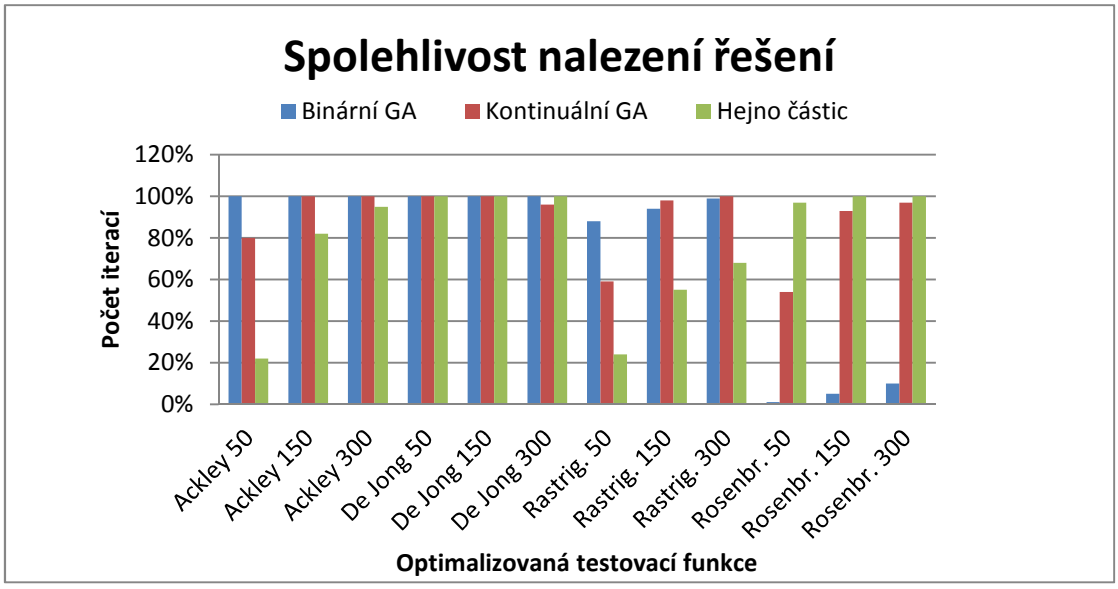
Funkce	Rastrig.	Rastrig.	Rastrig.	Rosenbr.	Rosenbr.	Rosenbr.
Vel. Populace	50	150	300	50	150	300
Prům. počet it.	46,77	41,04	39	35,23	24,82	20,09
Míra Q	53,14	43,66	39	3523	496,4	200,9
Spolehlivost R	88%	94%	99%	1%	5%	10%
STD účelové funkce	0,345562	0,067715	0,019908	0,185603	0,350335	0,350705

Spojitý GA

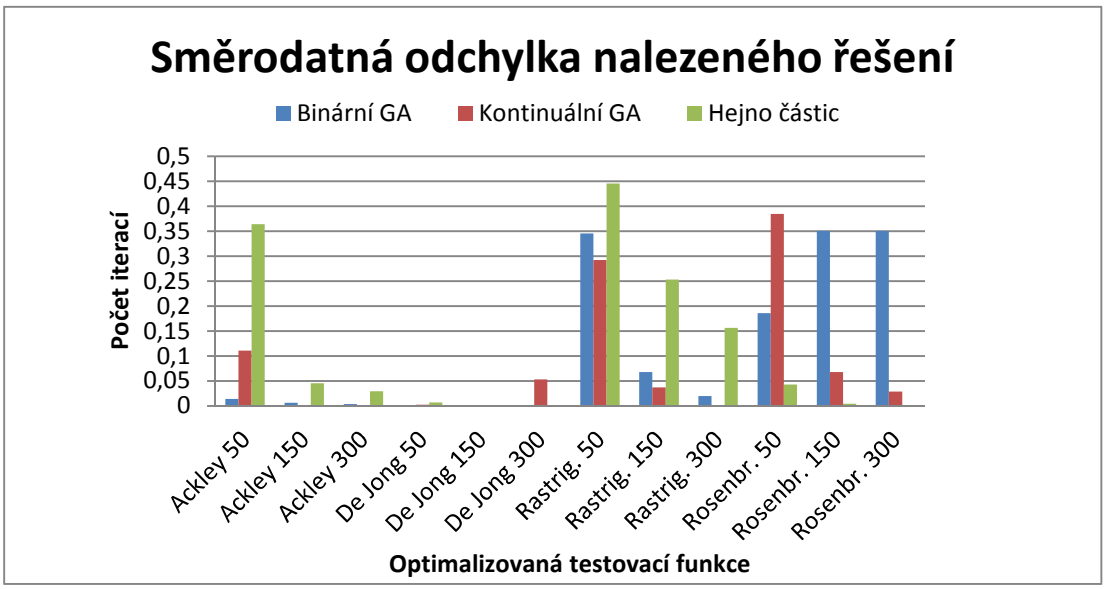
Vel. Populace	50	150	300	50	150	300
Prům. počet it.	40,2	55,13	53,75	37,33	44,98	44,43
Míra Q	42,62069	55,5	53,61458	39,03774	43,78889	44,96774
Spolehlivost R	59%	98%	100%	54%	93%	97%
STD účelové funkce	0,292317	0,036616	0,001713	0,384549	0,067792	0,028867

Hejno částic

Vel. Populace	50	150	300	50	150	300
Prům. počet it.	34,39	42,28	47,13	36,18	38,68	38,29
Míra Q	143,29	76,87	69,30	36,91	38,58333	38,42708
Spolehlivost R	24%	55%	68%	97%	100%	100%
STD účelové funkce	0,445607	0,252869	0,156105	0,042898	0,004286	0,001848



Obr. 20 Pravděpodobnost nalezení úspěšného řešení na testovací funkci
(číslo za názvem funkce označuje velikost populace)

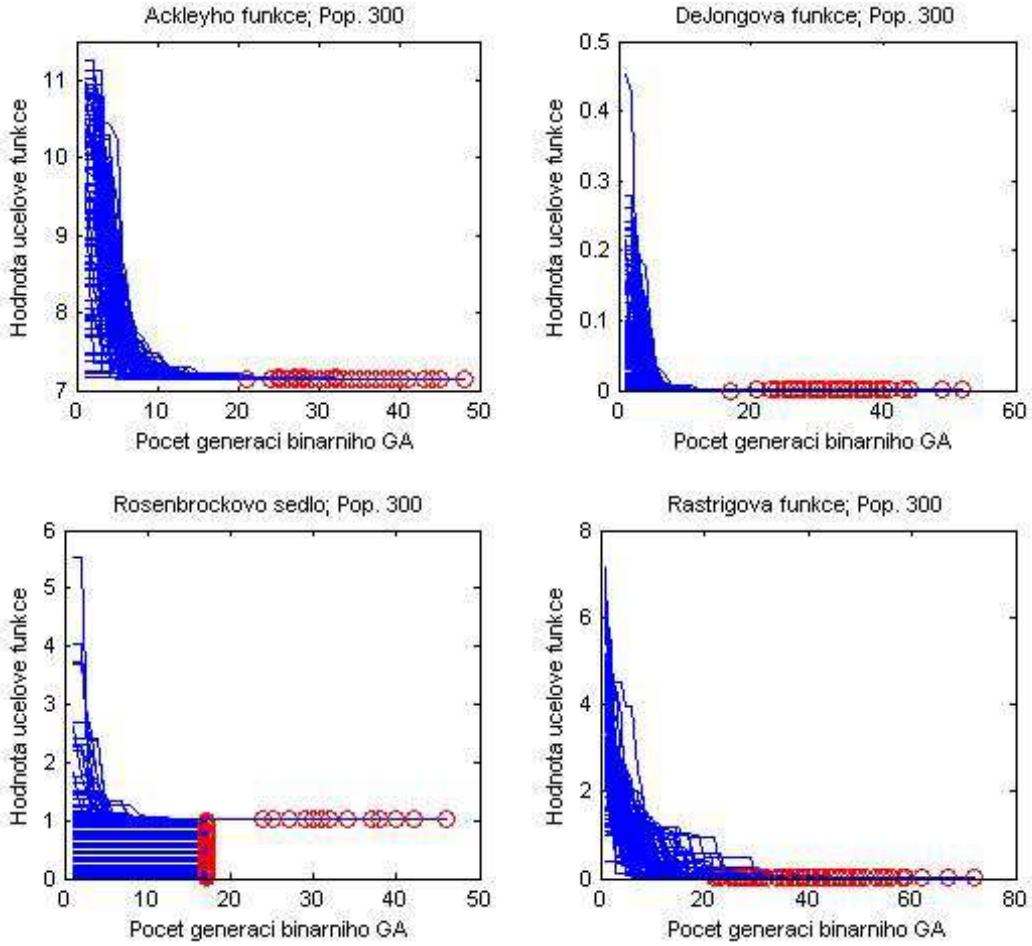


Obr. 21 Směrodatná odchylka nalezeného řešení
(číslo za názvem funkce označuje velikost populace)

Nejvíce spolehlivým algoritmem se ukázal dle výsledků zobrazených na obr. 20 spojitý genetický algoritmus, který dokázal najít správné řešení v průměru skrze všechny velikosti populací na 90%. Při velikosti populace 300 dosáhl spolehlivosti vůči nalezení dostatečně přesného řešení 98%.

Druhým algoritmem byl algoritmus hejna částic, který měl spolehlivost dosáhnutí úspěšného řešení skrze všechny velikosti populace průměrně 79%. Nejméně přesným algoritmem se ukázal být binární GA, u kterého byla spolehlivost nalezení řešení 75%.

S postupným zvětšováním počtu jedinců klesá směrodatná odchylka výsledných řešení jednotlivých algoritmů, jak ukazuje obr. 21. To znamená, že v případě větší populace je řešení přesnější.

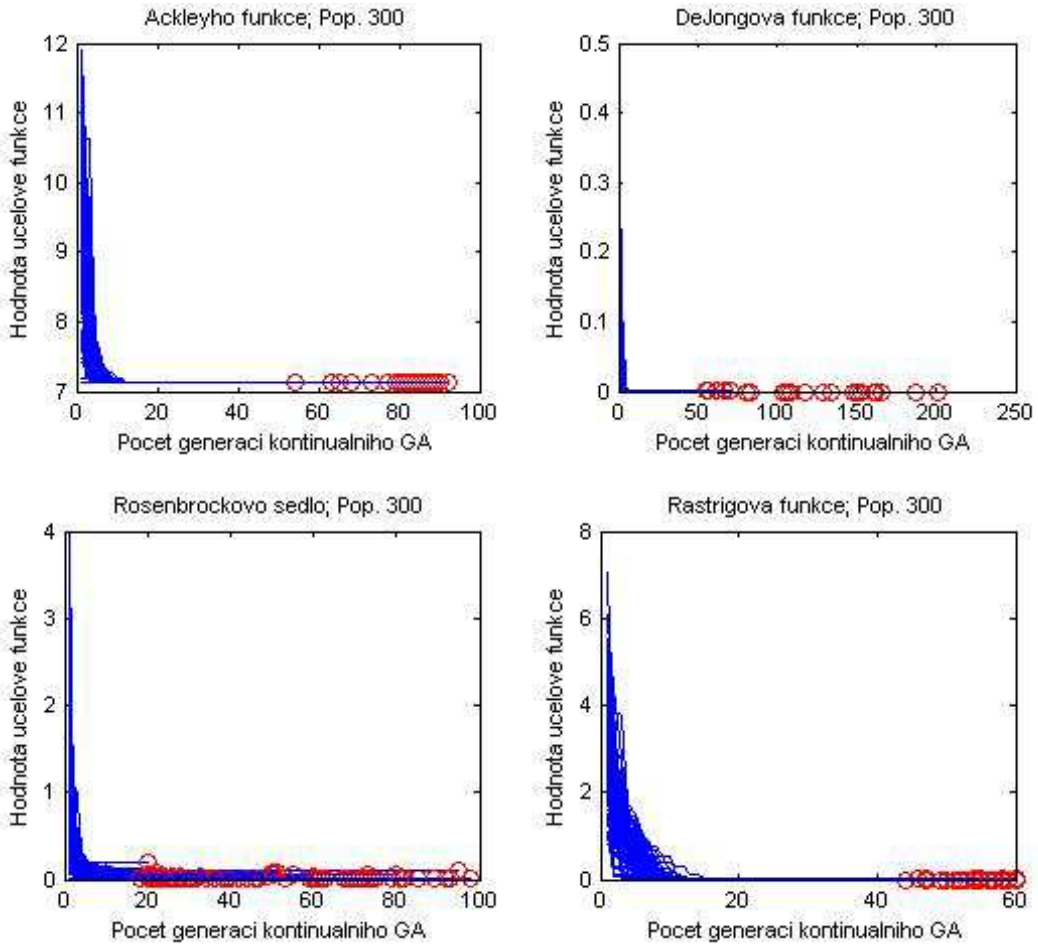


Obr. 22 Průběh konvergence binárního algoritmu na testovacích funkcích

Z výsledků je patrné, že nejúspěšnější v řešení testovacích funkcí byly algoritmy s populací o velikosti 300 jedinců. Tyto algoritmy jsou proto dále rozebrány z pohledu konvergence například na obr. 22, kde je modře zaznačen průběh jednotlivých konvergencí algoritmu a červeným kolečkem je zaznačeno nalezené řešení a ukončení algoritmu.

Z konvergencí na obr. 22 pro binární genetický algoritmus při populaci 300 jedinců lze pozorovat selhání algoritmu na Rosenbrockově testovací funkci. Selhání binárního algoritmu lze připisovat kvantování, které na jednu stranu urychluje konvergenci algoritmu, ale selhává v případě, že je globální minimum v prostoru řešení obklopeno hodnotami s velmi nízkým spádem.

Celkově lze na obr. 22 pozorovat díky kvantování i určitou míru segmentace, která se vyznačuje zachycením minimální hodnoty účelové funkce pro více generací. Tato vlastnost pomáhá algoritmu konvergovat rychleji, lze ale předpokládat, že by při zmenšení kritéria pro opakování nejmenší funkční hodnoty v jednotlivých generacích docházelo k značnému počtu neúspěšných řešení.

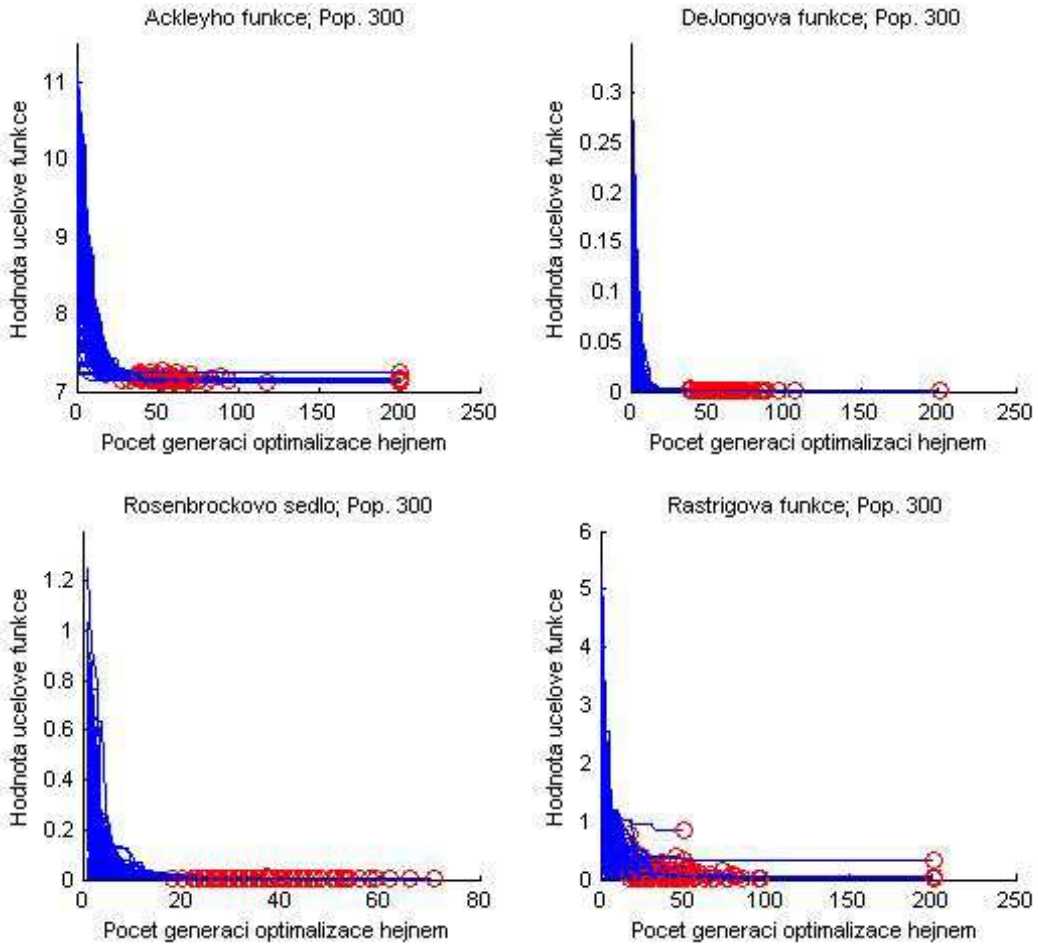


Obr. 23 Průběh konvergence spojitého algoritmu na testovacích funkcích

Obr. 23, který zobrazuje průběh konvergencí spojitého genetického algoritmu, ukazuje, že podobně jako u předchozích algoritmů je oblast globálního minima nalezena v průměru do 20. generace. Algoritmus však konverguje podstatně déle v okolí řešení, které je velmi blízké cílovému řešení, než je tomu například u binárního GA. Kritérium konvergence zde má zásadní dopad na počet kontrolního vyhodnocování účelové funkce.

Zajímavým aspektem v porovnání s binárním GA je určité shlukování jednotlivých řešení, které je patrné na Ackleho a Rastrigově funkci. Naopak průběh u De Jongovy funkce nebo průběh u Rosenbrockově funkci je velmi podobný binárnímu algoritmu. To je pravděpodobně způsobeno malým spádem, který se vyskytuje u globálního minima jednotlivých funkcí. Toto shlukování se projevuje přibližně stejným počtem iterací do konvergence.

Obr. 24 zobrazuje průběhy konvergence pro algoritmus hejna částic při populaci 300 jedinců na testovacích funkcích. Lze pozorovat, že nalezení optimální oblasti, kde může ležet globální minimum, je provedeno minimálně při stejné rychlosti jako u binárního algoritmu, dochází však, stejně jako v případě algoritmů předešlých, k velkým prodlevám před konvergencí.



Obr. 24 Průběh konvergence optimalizace hejnem částic na testovacích funkcích

Konvergence u algoritmu hejna částic probíhá více shlukově než u spojitého a binárního GA. Rovnoměrná distribuce řešení v rámci iterací byla pozorovatelná pouze na Rosenbrockově funkci. Shlukování pak bylo nejvíce pozorovatelné u Rastrigovy a Ackleyho funkci.

U Rastrigovy funkce jsou navíc výrazná dvě uvíznutí algoritmu v lokálních minimech, kdy v jednom případě došlo k předčasné konvergenci u 50. generace, v druhém případě dokonce konvergence neproběhla a algoritmus byl ukončen až maximálním počtem možných iterací.

Celkově z pohledu grafů konvergence na obr. 22, 23, 24 lze vyhodnotit jako nejlepší algoritmus optimalizaci hejnem díky rychlé a relativně spolehlivé konvergenci. Spojitý genetický algoritmus, který měl o poznání delší konvergenci na druhou stranu, dokázal nalézt řešení na všech testovacích funkcí.

Optimalizace hejnem částic a spojitý genetický algoritmus jsou na základě porovnání použity pro tvorbu hybridního algoritmu, který bude blíže představen v další části.

4 Hybridní algoritmus

Hybridní genetický algoritmus kombinuje výhody genetických algoritmů například s rychlostí lokálních optimalizačních procedur, popřípadě využívá diferenciace stochastických metod.

Genetické algoritmy excelují v hledání přibližné polohy globálního optima, jejich konvergence je však vůči tomuto optimu velmi pomalá, i pokud se jedná o jednoduchou kvadratickou funkci. Kombinace, kterých může hybridní genetický algoritmus nabývat:

1) Nechat genetický algoritmus do určité míry konvergovat a následně použít lokální metody
2) Zasadit do populace genetického algoritmu základní lokální minima
3) Po určitém počtu iterací přerušit genetický algoritmus, na nejlepších jedincích využít lokálního řešení a následně je navrátit do populace.

Hybridní algoritmus prezentovaný v diplomové práci kombinuje výše uvedené metody. K sestavení algoritmu je vybrána optimalizace hejnem a spojitý GA vzhledem k nejlepším výsledkům v předběžném porovnání.

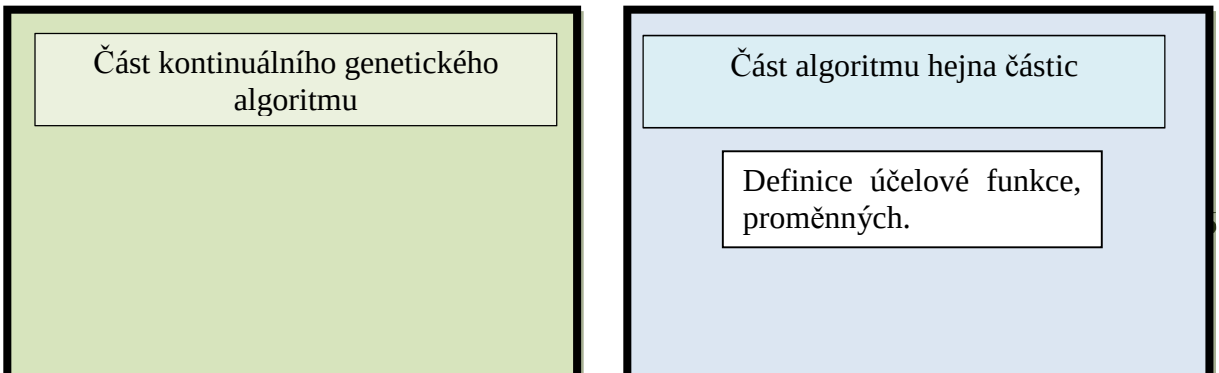
4.1 Hybridní GA-PSO

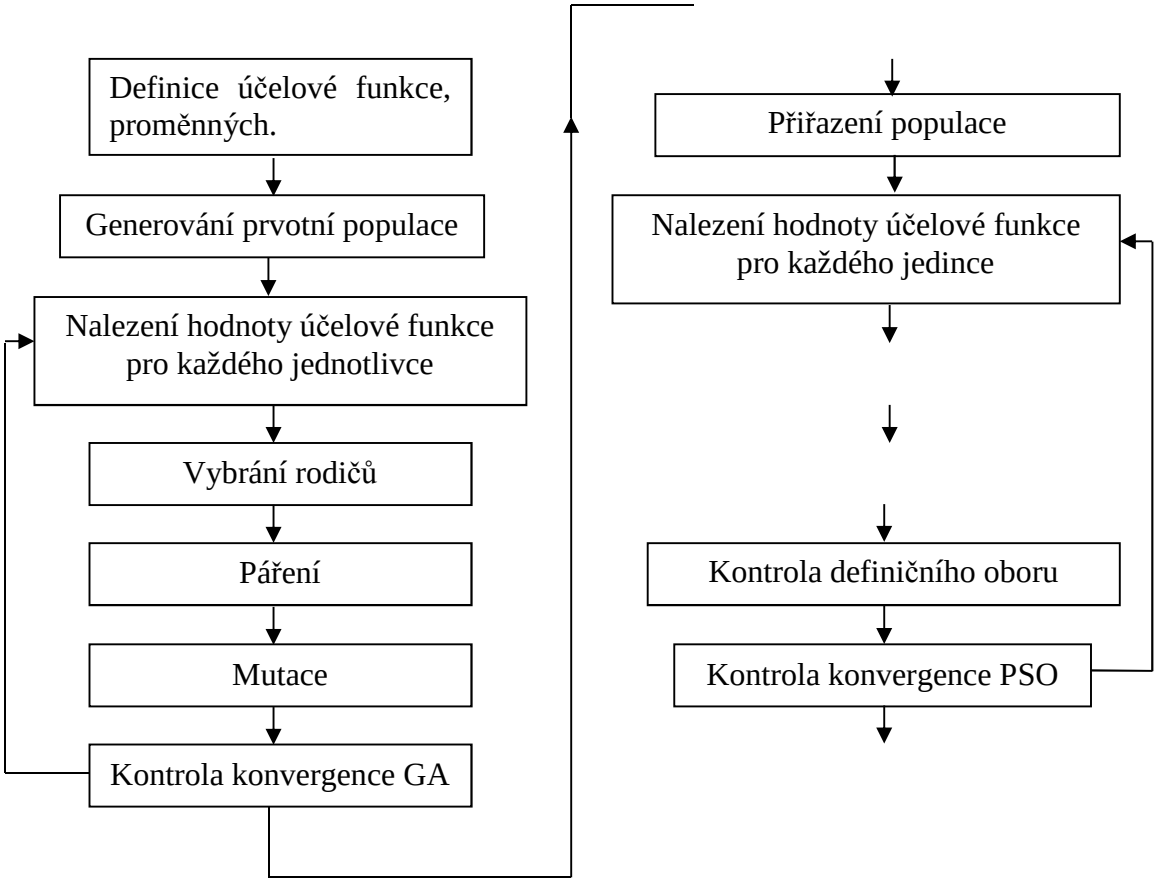
Hybridní algoritmus je sestaven na základě výsledků porovnání jednotlivých algoritmů v minulé kapitole ze spojitého genetického algoritmu a algoritmu hejna částic v tomto pořadí. Je použito holistického přístupu, kdy spojením dvou částí vznikne celek, který bude mít lepší výkonnost, než jednotlivé dvě části. U hybridního algoritmu se předpokládá, že diferenciací prohledávacích metod, lze zefektivnit hledání globálního minima, a to zvětšením pravděpodobnosti úspěšného hledání a snížením počtu iterací, tedy i evaluací nutných k nalezení globálního minima. Zmenšení počtu evaluací dané funkce je žádané hlavně tehdy, pokud je výpočet účelové funkce časově náročný.

Základní parametry jednotlivých metod zůstaly stejné. Pro spojitý GA byl zvolen stejný mutační poměr a stejný výběr jedinců. Hlavní změnou je zmenšení počtu iterací nutných pro ukončení výpočtové části spojitého GA. Kritérium konvergence bylo u spojitého GA sníženo z 15 na 8. Algoritmus svou část výpočtu zastaví, pokud nalezne minimum, které není překonáno po následujících 8 generací.

Toto minimum a jednotlivci, kteří prošli evolucí, jsou následně vloženi jako vstupní parametry pro optimalizaci hejnem části. U PSO došlo podobně jako u sGA ke snížení kritéria pro konvergenci z 15 na 7. Lze říci, že konvergenční kritérium předchozích algoritmů je do jisté míry podobné kritériu hybridního algoritmu. Počet iterací, kdy se dílčí minimum může v generacích algoritmu opakovat, pořád zůstává na hodnotě 15 iterací.

Testování hybridního GA-PSO algoritmu jasně ukazuje na lepší výsledky, než měly předchozí algoritmy, a to na všech testovacích funkcích.





Obr. 25 Schéma hybridního GA-PSO algoritmu

Testování dále ukázalo, že hybridní algoritmus dokázal najít řešení v daném intervalu na každé z testovacích funkcí se 100 % spolehlivostí a lepší výsledky dosáhl také v počtu iterací, potřebných pro nalezení daného řešení.

Schéma hybridního GA-PSO algoritmu vychází ze spojení dvou algoritmů. Počáteční populace pro celý algoritmus je náhodně generována a spojitý GA v této populaci hledá oblast lokálního minima, kterou určuje kritérium konvergence. V případě, že takovou oblast najde, předá populaci druhému algoritmu, který ji nadále zpracuje. Algoritmus se chová evolučně ve své první části a pohybově v části druhé.

Tabulka 15 ukazuje výsledky algoritmu na testovacích funkcích, kde hybridní GA-PSO dokázal nalézt řešení se 100 % spolehlivostí úspěšnosti na všech testovacích funkcích. Průměrný počet iterací na De Jongově funkci byl pouhých 25, což je výrazný posun, vzhledem k algoritmům prezentovaným dříve v práci.

Tab. 15 Statistika hybridniho GA-PSO algoritmu

Hybridní GA-PSO	Ackley	De Jong	Rastrig.	Rosenbr.
Vel. Populace	300	300	300	300
Prům. počet it.	37,52	24,63	35,91	31,93
Míra Q	37,52	24,63	35,91	31,93

Spolehlivost R	100%	100%	100%	100%
STD účelové funkce	0,005124	1,25E-05	0,002227	0,001396

Tab. 16 Porovnání jednotlivých algoritmů

	Hybrid GA-PSO	Binární GA	Spojitý GA	Hejno částic
Vel. Populace	300	300	300	300
Prům. počet it.	32,4975	30,8025	86,99	54,535
Míra Q	32,4975	40	88,76	59,92
Spolehlivost R	100%	77%	0,9825	91%
STD účelové funkce	0,00219	0,093474	0,02084	0,046882

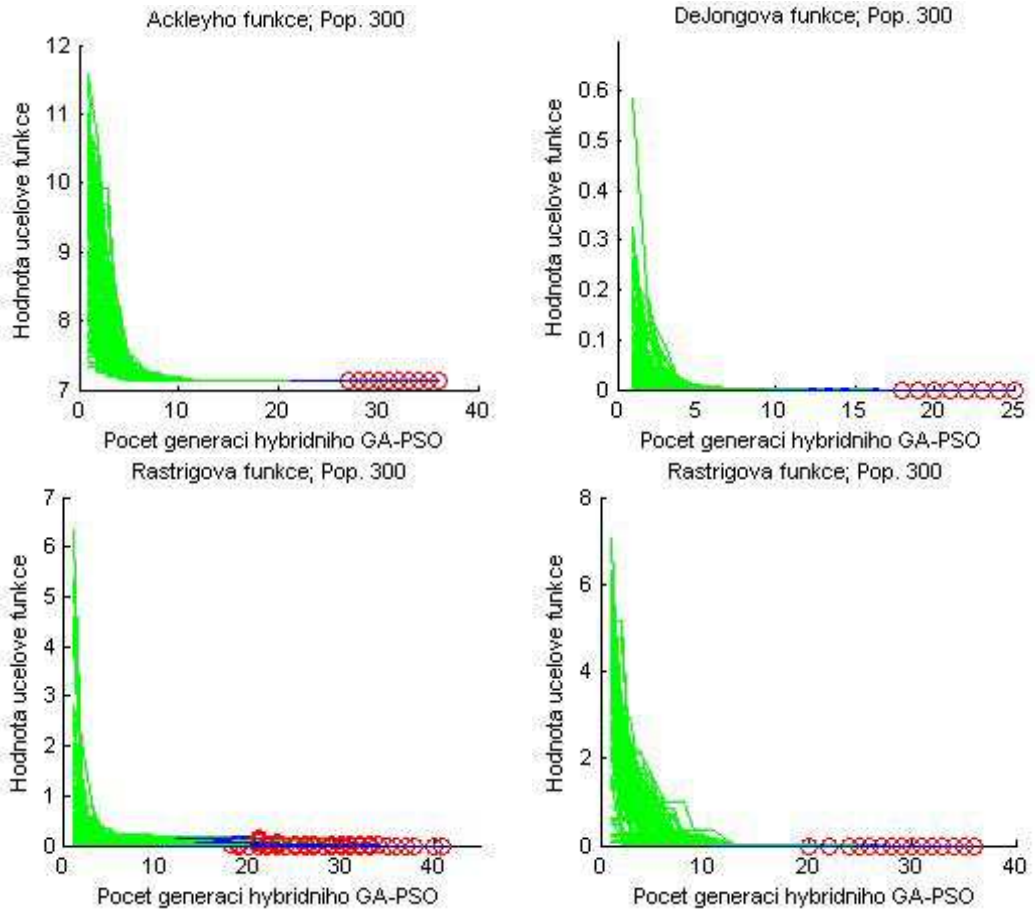
Celková statistika algoritmu skrze všechny testovací funkce je následně porovnávána s výslednými hodnotami ostatních algoritmů. Tabulka 15 uvádí průměrné naměřené hodnoty u algoritmů na testovacích funkcích při velikosti populace 300 jedinců v každém jednotlivém algoritmu.

Binárnímu algoritmu trvalo nalézt řešení v průměru o téměř dvě generace méně, avšak pravděpodobnost, že toto řešení leželo v intervalu řešení, byla pouhých 77%. Hybridnímu algoritmu trvalo nalézt řešení s konvergencí v průměru 32,5 iterací, což znamená více než dvojnásobné zrychlení oproti sGA a jednonásobné zrychlení oproti PSO.

Algoritmus dále exceloval v přesnosti daných řešení, kdy je průměrná směrodatná odchylka o řád menší než u ostatních algoritmů. Při aplikaci porovnávací míry Q, je podle výsledků v tab. 15 nejvýkonnějším algoritmem hybridní GA-PSO s hodnotou 32,49.

Konvergenční grafy zobrazené v Grafu 4 lépe ilustrují spojení dvou metod. Spojitý GA je zaznačeno zeleně, PSO modře, a konečné řešení pak červeným kroužkem.

Na obr. 26 je znázorněna krátká doba konvergence, která je také uváděná v tab. 16. Je zřejmé zlepšení vůči předchozím algoritmům. Všechny funkce (až na Rastrigovu) zvládl algoritmus bez větších problémů. V grafech i v tab. 15 vidíme, že směrodatná odchylka řešení byla velmi malá. Algoritmus tvořil větší shluk a měl větší variabilitu řešení na Rastrigově funkci. To má opodstatnění ve vysoké náročnosti této funkce.



Obr. 26 Průběh konvergence hybridního GA-PSO na testovacích funkcích

Algoritmus celkově ukázal, že diferenciace metod v jednom algoritmu může vést k podstatnému zlepšení konvergence a rychlosti výpočtu. Otázkou však zůstává, zdali toto zlepšení bylo dáno tím, že sGA dokáže relativně rychle identifikovat místo, kde se nachází globální minimum, a PSO může sloužit k rychlé finální optimalizaci, nebo zdali lze výměnou těchto dvou metod dosáhnout lepšího výsledku.

4.2 Hybridní PSO-GA

Hybridní PSO-GA algoritmus je v diplomové práci vytvořen díky velmi pozitivním výsledkům předchozího GA-PSO. V tomto případě je hlavním kritériem sledování podobnost jednotlivých porovnávacích parametrů u každé z metod

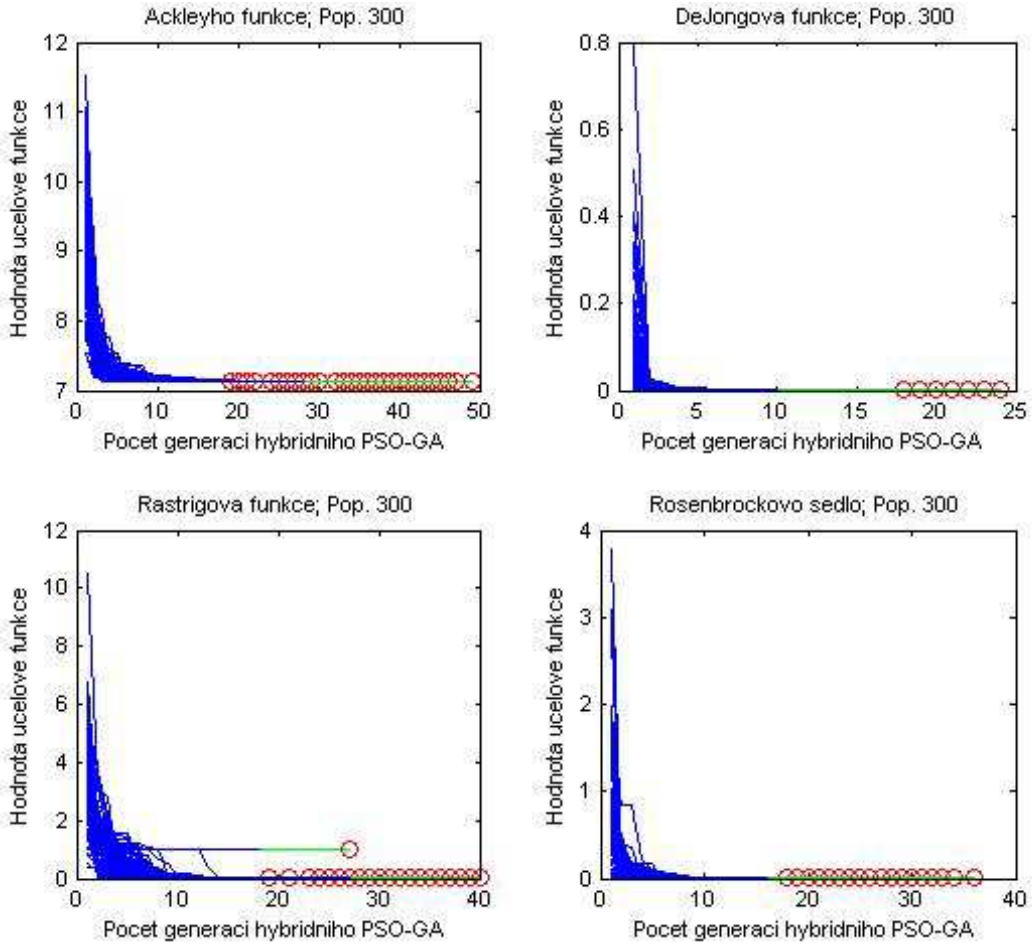
Při tvorbě algoritmu PSO-GA zůstaly parametry u obou částí stejné. Rozdíly byly jenom v konvergenčním kritériu, které bylo pro hejno nastaveno na hodnotu 8 a u spojitého GA bylo nastaveno na hodnotu 7.

Jednotlivé dílčí výsledky hybridního PSO-GA na jednotlivých funkcích uvádí tab. 17. Zde je patrné výrazné zvýšení výkonnosti oproti testovaným algoritmům již na jednotlivých funkcích. Celkové výsledky algoritmu pak vychází, podobně jako výsledky GA-PSO, výrazně lepší, než výsledky jednotlivých metod (bGA,sGA,PSO) jak následně uvádí tab. 18. Je vidět i

podstatné zlepšení v průměrném počtu iterací vzhledem k GA-PSO. Zhoršení nastalo ve spolehlivosti nalezení úspěšného řešení, které kleslo oproti GA-PSO na hodnotu 99%.

Tab. 17 Statistika PSO-GA na testovacích funkcích

Hybridní PSOGA	Ackley	De Jong	Rastrig.	Rosenbr.
Vel. Populace	300	300	300	300
Prům. počet iterací.	34,6	20,44	30,88	25,61
Q	34,6	20,44	30,88	25,61
R	99%	100%	99%	99%
STD účelové funkce	0,005124	1,25E-05	0,002227	0,001396



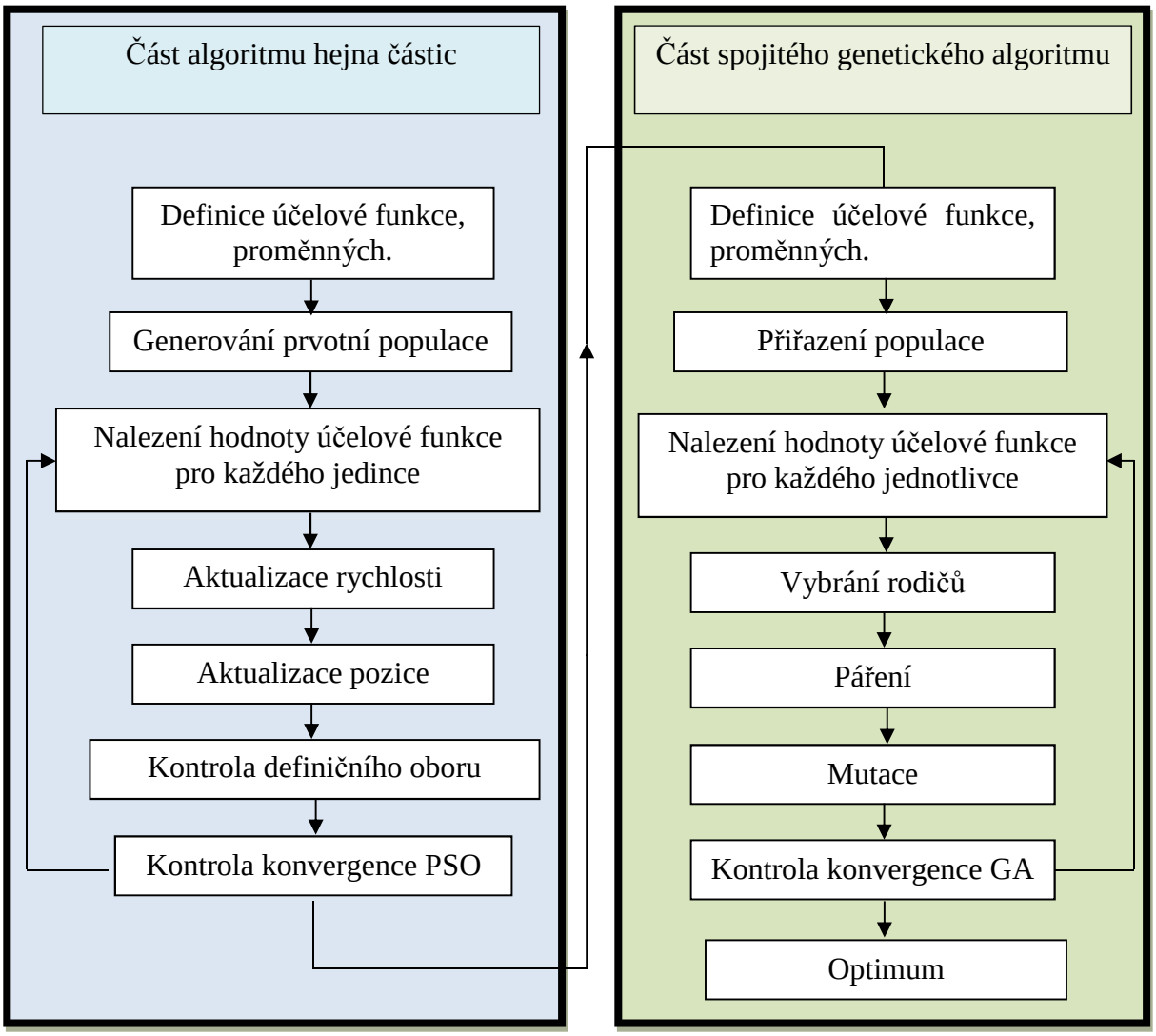
Obr. 27 Průběh konvergence hybridního PSO-GA algoritmu na testovacích funkcích

Porovnání s ostatními algoritmy ukazuje tab. 17. Konvergenční grafy zobrazené v obr. 26 ilustrují spojení dvou metod (PSO-GA). PSO je zaznačeno modře, spojitý GA je zaznačeno zeleně a konečné řešení pak červeným kroužkem.

Testovací funkce většinou netvořily algoritmu výrazné problémy. Jedinou znatelnou chybou algoritmu bylo uvíznutí v lokálním minimu Rastrigovy funkce při optimalizaci hejnem, kdy genetický algoritmus už nedokázal najít vhodnější řešení. Znatelné zrychlení lze vidět na testovací funkci Rosenbrokova sedla, kdy průměrný počet iterací klesl na 25,61, což je nejnižší hodnota ze všech prezentovaných přístupů.

Tab. 18 Porovnání celkových výsledků jednotlivých algoritmů

	Hybrid PSO-GA	Hybrid GA-PSO	Binární GA	Spojitý GA	Hejno částic
Vel. Populace	300	300	300	300	300
Prům. počet it.erací	27,882	32,4975	30,8025	86,99	54,535
Míra Q	28,16	32,4975	40	88,58	59,92
Spolehlivost R	99%	100%	77%	98,2%	91%
STD účelové funkce	0,00259	0,00219	0,093474	0,02084	0,046882



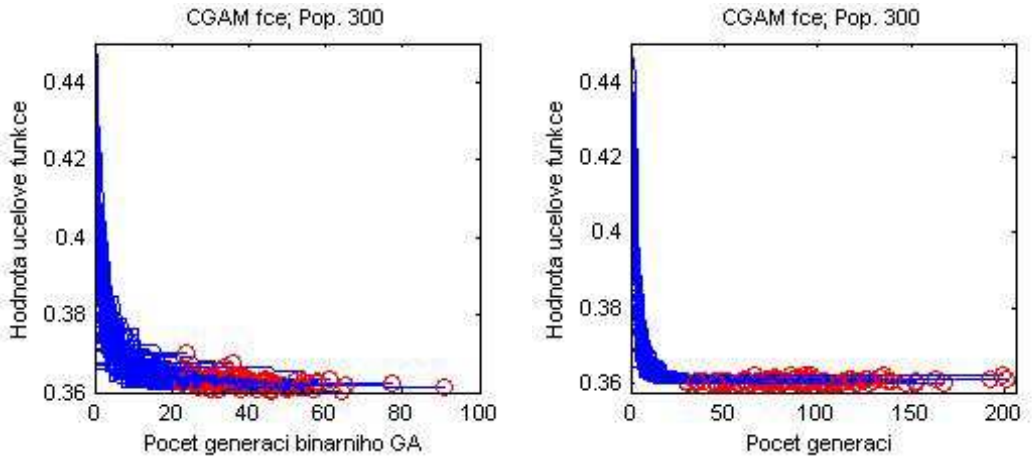
Obr. 28 Schéma hybridního PSO-GA algoritmu

PSO-GA algoritmus vychází z opačného spojení dvou algoritmů. Počáteční populace pro celý algoritmus je náhodně generována a PSO v této populaci hledá oblast lokálního minima, kterou určuje kritérium konvergence. V případě, že takovou oblast najde, předá populaci druhému algoritmu, sGA, který ji dále zpracuje. Algoritmus se chová pohybově ve své první polovině a evolučně v polovině druhé.

5 Optimalizace CGAM

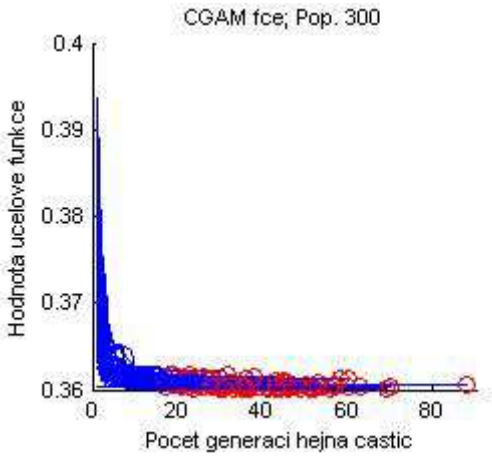
Problém optimalizace představeného kogeneračního systému je řešen na konci práce vzhledem k faktu, že je samoúčelné stavět algoritmus pro vyhledávání globálního minima pouze na účelovou funkci CGAM problému, vzhledem k tomu že se prostor řešení může výrazně lišit u více komplexnějších problémů.

Rovnice pro výpočet nákladů kogeneračního systému byla řešena všemi představenými algoritmy s předpoklady, že jejich chování bude podobné, jako bylo v předchozích testech. Tento předpoklad lze postavit na výsledcích jednotlivých algoritmů u lehčí Ackleyho a těžší Rastrigově funkci, které obsahují mnoho lokálních minim s rozdílným spádem funkční hodnoty okolí.



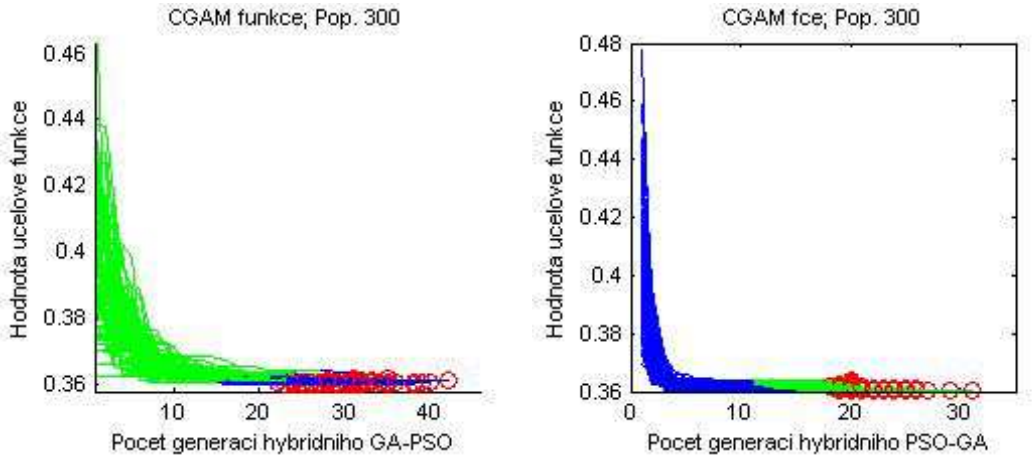
Obr. 29 Průběh konvergence na CGAM problému pro binární a spojitý GA

Při pohledu na obr. 29 vidíme, že základní algoritmy se chovají velmi podobně, jako při prvotním testování. I když je rychlost binárního genetického algoritmu větší, jeho řešení nejsou tak přesná. Naopak spojitý genetický algoritmus má velmi přesná řešení, která jsou vykoupena dlouhou dobou potřebnou ke konvergenci.



Obr. 30 Průběh konvergence na CGAM problému pro hejno částic GA

Průběh konvergence optimalizace hejnem, která je na obr. 30 výrazně překonává binární i spojitý GA algoritmy v porovnání rychlosti konvergence a přesnosti řešení.



Obr. 31 Průběh konvergence na CGAM problému pro hybridního GA-PSO a PSO-GA algoritmů

Oba dva hybridní algoritmy, které jsou zobrazeny na obr. 31, mají na účelové funkci CGAM problému lepší výsledky, než algoritmy ze kterých se vycházelo, a to jak v rychlosti, tak v přesnosti řešení. Hybridní PSO-GA tuto skutečnosti posunuje ještě dále, vzhledem k tomu, že průměrná rychlost pro nalezení velmi přesného výsledku je zde o téměř 9 iterací rychlejší, než u hybridní verze GA-PSO.

Výsledky optimalizace CGAM funkce z pohledu stavových optimalizovaných hodnot jsou uvedeny níže v tabulce tab. 19.

Tab. 19 Výsledky algoritmů pro CGAM optimalizaci

	Hybrid PSO GA	Hybrid GA PSO	Binární GA	Spojitý GA	Hejno částic
Prům. počet it.	21,83838384	30,59596	36,31313	91,19192	37,45455
STD účelové funkce	0,000876253	0,000337	0,00156	0,000738	0,000453
Minimální hodnota	0,360183583	0,359966	0,360472	0,359997	0,360031
Maximální hodnota	0,364588821	0,361559	0,370324	0,363576	0,362087

Ve výsledcích optimalizace CGAM nebyl záměrně zahrnut výpočet spolehlivosti nalezení úspěšného řešení. Z hlediska nejlepšího řešení totiž všechny algoritmy dokázaly najít velmi podobná řešení. Interval, který určuje, od které hranice je řešení proveditelné, závisí na ostatních ekonomických faktorech.

Výsledná nejlepší řešení stavových proměnných CGAM účelové funkce jsou spolu s příslušným algoritmem uvedena v tab. 20.

Tab. 20 Výsledné stavové proměnné CGAM problému pro nejlepší výsledek

	Hybrid PSO GA	Hybrid GA PSO	Binární GA	Spojitý GA	Hejno částic
Náklady	0,360184	0,359966	0,360472	0,359997	0,360031
P_2/P_1	8,853203	8,546915	8,94	8,720261	8,755911
η_{AC}	0,842995	0,845593	0,854	0,847771	0,847404
T_3	916,6248	920,9043	909,5	913,9309	882,8948
η_{GT}	0,877655	0,876919	0,8884	0,878334	0,875214
T_4	1491,284	1492,627	1498,8	1490,164	1491,092

Při porovnání výsledků, které byly předloženy např. [2], [3], [9], lze konstatovat, že algoritmy uvedené v diplomové práci došly k výsledkům lepším, než přístupy uvedené ve studiích.

Náklady zobrazené v tab. 20, které jsou velmi důležitým kvalitativním kritériem, se podařilo v případě hybridního GA-PSO algoritmu snížit o 0,5% oproti originálnímu řešení CGAM problému.

6 Závěr

V diplomové práci byly představeny heuristické metody, které dokáží efektivně optimalizovat složité oblasti řešení pro nalezení globálního optima. Na základě těchto metod byly představeny dva hybridní algoritmy, které jsou různou kombinací algoritmů, sestavené na základě testů jejich jednotlivých výkonností pro různě obtížné funkce. Kombinace těchto algoritmů byla dána předpokladem, že diferenciace prohledávacích metod v jednom algoritmu, může mít pozitivní dopad na celkovou výkonnost a rychlost algoritmu.

Lze říci, že všechny zde představené algoritmy vedly k uspokojujícím výsledkům optimalizace. Lze ale poukázat, že hybridní metody představené v práci se ukázaly jako výkonnější při hledání globálního minima, než jednotlivé metody optimalizace hejna, binárního či spojitého genetického algoritmu. Algoritmus, který však exceloval, byl hybridní algoritmu PSO-GA, který se v daném prostoru řešení chová pohybově podle vzoru optimalizace hejnem částic a při nalezení dostatečného řešení změní své chování na evoluční podle vzoru spojitého genetického algoritmu.

Metoda diferenciace prohledávacích metod tedy vedla k více než uspokojivým výsledkům a lze ji doporučit pro další optimalizace energetických systémů.

Bibliografie

[1]. **MICHALEWICZ,** Z and D B FOGEL. How to Solve It: Modern Heuristics. B.m.: Springer, 2004. ISBN 9783540224945. .

[2]. **VALERO,** Antonio, Miguel A LOZANO, Luis SERRA, George TSATSARONIS, Javier PISA, Christos FRANGOPOULOS and Michael R VON SPAKOVSKY. {CGAM} problem: Definition and conventional solution. Energy [online]. 1994, vol. 19, no. 3, str. 279–286. ISSN 0360-5442..

[3]. **PADILHA,** Ricardo S, Hugo F L SANTOS, Marcelo J COLACO and Manuel E CRUZ. Single and Multi-Objective Optimization of a Cogeneration System Using Hybrid Algorithms. HEAT TRANSFER ENGINEERING [online]. 2009, vol. 30, no. 4, str. 261–271. ISSN 0145-7632. Re.

[4]. **MASTNÝ,** Petr. Malé zdroje elektrické energie, Přednáška. Brno:VUT, [cit. 2014-03-03] dostupné z: http://http://www.ueen.feec.vutbr.cz/~mastny/vyuka/mmze/prednasky/07_08_pr.pdf.

[5]. **TRÁVNÍČEK,** Petr. Kogenerační jednotka GE Jenbacher [online] [cit. 2014 04-04] dostupné z:http://biom.cz/cz/obrazek/kogeneracni-jednotka-ge-jenbacher-2 .

[6]. **KNOPF,** F C. Modeling, Analysis and Optimization of Process and Energy Systems. B.m.: Wiley, 2011. ISBN 9781118121146. .

[7]. **FRANGOPOULOS,** Christos A. Astrlication of the thermoeconomic functional astrroach to the {CGAM} problem. Energy [online]. 1994, vol. 19, no. 3, str. 323–342. ISSN 0360-5442. Dostupné z: doi:http://dx.doi.org/10.1016/0360-5442(94)90114-7.

[8]. **VALERO,** A, M A LOZANO, L SERRA and C TORRES. Astrlication of the exergetic cost theory to the {CGAM} problem. Energy [online]. 1994, vol. 19, no. 3, str. 365–381. ISSN 0360-5442. Dostupné z: doi:http://dx.doi.org/10.1016/0360-5442(94)90116-3.

[9]. **TSATSARONIS,** George and Javier PISA. Exergoeconomic evaluation and optimization of energy systems — astrlication to the {CGAM} problem. Energy [online]. 1994, vol. 19, no. 3, str. 287–321. ISSN 0360-5442. Dostupné z: doi:http://dx.doi.org/10.1016/0360-5.

[10]. **MAŇAS,** M. Optimalizační metody. In: . B.m.: SNTL, 1979, Teoretická knižnice inženýra. .

[11]. **MATEMATIKA ONLINE,** Lokální, vázané a globální extrémy, Studijní text, Brno: VUT, 2005, [online] [cit 2014 03-20] dostupné z: http://mathonline.fme.vutbr.cz/Lokalni-vazane-a-globalni-extremy/sc-97-sr-1-a-98/default.aspx.

[12]. —, Lokální, vázané a globální extrémy, Studijní text, Brno: VUT, 2005, [online] [cit 2014 03-20] dostupné z: http://mathonline.fme.vutbr.cz/Lokalni-vazane-a-globalni-extremy/sc-97-sr-1-a-98/default.aspx.

[13]. **WOLPERT,** David H and William G MACREADY. No free lunch theorems for optimization. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. 1997, vol. 1, no. 1, str. 67–82. .

[14]. **HAUPT,** R L and S E HAUPT. Practical Genetic Algorithms. B.m.: Wiley, 2004. Wiley InterScience electronic collection. ISBN 9780471671756. .

[15]. **HOLLAND,** J H. Adaptation in natural and artificial systems: an introductory analysis with astrlications to biology, control, and artificial intelligence. B.m.: University of Michigan Press, 1975. ISBN 9780472084609. .

[16]. **GOLDBERG,** D E. Genetic Algorithms in Search, Optimization, and Machine Learning. B.m.: Addison-Wesley, 1989. Artificial Intelligence. ISBN 9780201157673. .

[17]. **STRACHOTA,** Pavel. Teorie signálu pro počítačovou grafiku, Přednáška. Praha: ČVUT, [cit. 2014-03-03] dostupné z: http://saint-paul.fjfi.cvut.cz/base/sites/default/files/POGR/POGR2/02.teorie_signalu.pdf.

[18]. **RADCLIFFE,** Nicholas J. Forma Analysis and Random Respectful Recombination. In: ICGA. 1991, p. 222–229. .

[19]. **KENNEDY,** James, Russell EBERHART and OTHERS. Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks. 1995, p. 1942–1948. .

[20]. **TVRDÍK,** Josef. Stochastické algoritmy pro globální optimalizaci, Skripta. Ostrava: Ostravská univerzita v Ostravě, [cit. 2014-03-03] dostupné z: http://www1.osu.cz/~tvrdik/down/files/STAGO_10.pdf.

[21]. **MOLGA,** Marcin; SMUTNICKI, Czesław. Test functions for optimization needs. Test functions for optimization needs, 2005..

[22]. **CLERC,** M and J KENNEDY. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. Evolutionary Computation, IEEE Transactions on [online]. 2002, vol. 6, no. 1, str. 58–73. ISSN 1089-778X. Dostupné z: doi:10.1109/42.

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha 1 – Účelová funkce

```
%% OPTIMALIZACNI FUNKCE

function [y]=optimal_fceC(abc)

[rozmer1,~]=size(abc);
for i=1:rozmer1

    P2=(27-7)*abc(i,1)/1000+7;
    etaAC=(0.9-0.7)*abc(i,2)/1000+0.7;
    T3=(1100-600)*abc(i,3)/1000+600;
    etaGT=(0.9-0.7)*abc(i,4)/1000+0.7;
    T4=(1500-1100)*abc(i,5)/1000+1100;


  y(i)=0.00002444993056*(-(10276*log((T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) +
27384/(35100.0/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) ...
      - 1.17*T4 + (294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) -
(1.17*((30000*(1.17*T4 - 348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008...
      - 1) + 1) - 1.17*T4 + (294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) -
(30000*(1.004*T3 - 299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008...
      - 1) + 1) - 1.17*T4 + (294.5245364*((0.9871668312*P2)^0.2857142857 -
1))/etaAC))))/(1.17*T4 - 49348.8355)) + (30000*((299.3426*((0.9871668312*P2)^0.2857142857 -
1))/etaAC - 1.004*T3 + 299.3426))/(((1.004*T3...
      - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) + 1)*(35100.0/(((1.004*T3 - 1.17*T4 +
49.4929)/(1.17*T4 - 49348.8355) + 1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) +
1) - 1.17*T4 + (294.5245364*((0.9871668312*P2)^0.2857142857...
      - 1))/etaAC)) - (1.17*((30000*(1.17*T4 - 348.8355))/(((1.004*T3 - 1.17*T4 +
49.4929)/(1.17*T4 - 49348.8355) + 1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) +
1) - 1.17*T4 + (294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))...
      - (30000*(1.004*T3 - 299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355)
+ 1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4...
      - 49348.8355))*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) -
470.52)/(T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) + 37660/(35100.0/(((1.004*T3...
      - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (1.17*((30000*(1.17*T4 -
348.8355))/(((1.004*T3...      - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 ...
      + 49.4929)/(1.17*T4 - 49348.8355) + 1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008
- 1) + 1) - 1.17*T4 + (294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 -
49348.8355)) + (30000*((299.3426*((0.9871668312*P2)^0.2857142857...
      - 1))/etaAC - 1.004*T3 + 299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 -
49348.8355) + 1)*(35100.0/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 ...
      + (294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (1.17*((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4...
      + (294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 -
49348.8355))*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) -
298.15)))/(10276/(35100.0/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (1.17*((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 - 49348.8355)) +
172.37))^0.8 + 0.00001533981944*(-(1950000.0*log(-(T3 -
T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) +
1))/(T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) -
(298.15*((0.9871668312*P2)^0.2857142857 - 1))/etaAC +
```

```
(30000*((299.3426*((0.9871668312*P2)^0.2857142857 - 1))/etaAC - 1.004*T3 +
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) + 1)*(35100.0/(((1.004*T3 -
1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (1.17*((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 -
49348.8355))*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - 298.15))*(30000/(((1.004*T3 -
1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - ((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)))/(1.17*T4 -
49348.8355))*((299.3426*((0.9871668312*P2)^0.2857142857 - 1))/etaAC - 1.004*T3 +
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) + 1)*(35100.0/(((1.004*T3 -
1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) + )*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008
- 1) + 1) - 1.17*T4 + (294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) -
(1.17*((30000*(1.17*T4 - 348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 -
49348.8355))*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)*(T3 -
(298.15*((0.9871668312*P2)^0.2857142857 - 1))/etaAC +
(30000*((299.3426*((0.9871668312*P2)^0.2857142857 - 1))/etaAC - 1.004*T3 +
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) + 1)*(35100.0/(((1.004*T3 -
1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) + )*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008
- 1) + 1) - 1.17*T4 + (294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) -
(1.17*((30000*(1.17*T4 - 348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 -
49348.8355))*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - 298.15)))^0.6 +
(0.2*((30000*(1.17*T4 - 348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 - 49348.8355) +
0.000004407686111*(((30000*(1.17*T4 - 348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 -
49348.8355) + 1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)))/(1.17*T4 - 49348.8355) -
30000/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)))^1.2 + 0.00002444993056*(-
(27384*log((T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) +
(30000*((299.3426*((0.9871668312*P2)^0.2857142857 - 1))/etaAC - 1.004*T3 +
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) + 1)*(35100.0/(((1.004*T3 -
1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (1.17*((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 -
49348.8355))*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) -
485.52)/(T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) + 27384/(35100.0/(((1.004*T3 -
```

```
1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (1.17*((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 - 49348.8355)) +
(30000*((299.3426*((0.9871668312*P2)^0.2857142857 - 1))/etaAC - 1.004*T3 +
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) + 1)*(35100.0/(((1.004*T3 -
1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (1.17*((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 -
49348.8355))*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) -
470.52)))/(27384/(35100.0/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (1.17*((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 - 49348.8355)) +
15.0))^0.8 - (0.114322963*(exp(0.018*T4 - 26.4) + 1))/(((1.004*T3 - 1.17*T4 +
49.4929)/(1.17*T4 - 49348.8355) + 1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) +
1) - 1.17*T4 + (294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) +
(0.000000006698611111*log(0.820980997*P2)*(7989000.0/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4
- 49348.8355) + 1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (266.3*((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 -
49348.8355))*exp(0.036*T4 - 54.4) + 1))/(etaGT - 0.92) +
(0.007835986344*P2*log(0.9871668312*P2))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 -
49348.8355) + 1)*(etaAC ...
      - 0.9)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) + 0.001108486167;


  p(i)=T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) + 27384/(35100.0/(((1.004*T3 -
1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (1.17*((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 - 49348.8355)) +
(30000*((299.3426*((0.9871668312*P2)^0.2857142857 - 1))/etaAC - 1.004*T3 +
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) + 1)*(35100.0/(((1.004*T3 -
1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (1.17*((30000*(1.17*T4 -
348.8355))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - (30000*(1.004*T3 -
299.3426))/(((1.004*T3 - 1.17*T4 + 49.4929)/(1.17*T4 - 49348.8355) +
1)*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC))))/(1.17*T4 -
49348.8355))*(1.17*T4*(etaGT*(1/(0.820980997*P2)^0.2481203008 - 1) + 1) - 1.17*T4 +
(294.5245364*((0.9871668312*P2)^0.2857142857 - 1))/etaAC)) - 485.52;
    if not(isreal(y(i))) || p(i)<=0 || y(i)<0
        y(i)=10e7;
    end
end
```

Příloha 2 – Funkce na převod binárních proměnných

```
function [cislo]=bin2normal(B,aa,bb,cc,dd, ee)
[radek,~]=size(B);
 for j=1:radek

bits1=B(j,(1:aa));
bits2=B(j,(aa+1):(aa+bb));
bits3=B(j,(aa+bb+1):(aa+bb+cc));
bits4=B(j,(aa+bb+cc+1):(aa+bb+cc+dd));
bits5=B(j,(aa+bb+cc+dd+1):(aa+bb+cc+dd+ee));

[~,N]=size(bits1);
k=N;P=0;
for i=1:k
    P(i)=(bits1(1,i)*(2^(k-i)));
end
cislo(j,1)=sum(P);

[M,N]=size(bits2);
k=N;P=0;
for i=1:k
    P(i)=(bits2(1,i)*(2^(k-i)));
end
cislo(j,2)=sum(P);

[M,N]=size(bits3);
k=N;P=0;
for i=1:k
    P(i)=(bits3(1,i)*(2^(k-i)));
end
cislo(j,3)=sum(P);

[M,N]=size(bits4);
k=N;P=0;
for i=1:k
    P(i)=(bits4(1,i)*(2^(k-i)));
end
cislo(j,4)=sum(P);
[M,N]=size(bits5);
k=N;P=0;
for i=1:k
    P(i)=(bits5(1,i)*(2^(k-i)));
end
cislo(j,5)=sum(P);
end
end
```

Příloha 3 – Binární genetický algoritmus

```
%%%BINARNI GENETICKY ALGORITMUS
%% Algoritmus byl vytvoren na zaklade literatury uvedene v bibliografii a za pomoci studii
clear all
for por=1:100
clear minc
tic
ff='optimal_fceB';
pocBits=[10,10,10,10,10];
ncasticebits=sum(pocBits);
varhi=1000; varlo=0;
maxit=200;
velikost_popuplace=300;
mutrate=.1;
selekce=0.5;
Nt=ncasticebits;        keep=floor(selekce*velikost_popuplace);
poc_iteraci=0;
pop=round(rand(velikost_popuplace,Nt));
castice=bin2normal(pop,pocBits(1),pocBits(2),pocBits(3),pocBits(4),pocBits(5));
hodn_ucel=feval(ff,castice);
[hodn_ucel,ind]=sort(hodn_ucel);
castice=castice(ind,:);pop=pop(ind,:);
minc=min(hodn_ucel);
meanc=mean(hodn_ucel);
while poc_iteraci<maxit
poc_iteraci=poc_iteraci+1;
%parovani
M=ceil((velikost_popuplace-keep)/2);
prob=flipud([1:keep]'/sum([1:keep]));
pravdeps=[0 cumsum(prob(1:keep))'];
vyber1=rand(1,M);
vyber2=rand(1,M);
ic=1;
while ic<=M
    for id=2:keep+1
        if vyber1(ic)<=pravdeps(id) && vyber1(ic)>pravdeps(id-1)
            ma(ic)=id-1;
        end % if
        if vyber2(ic)<=pravdeps(id) && vyber2(ic)>pravdeps(id-1)
            pa(ic)=id-1;
        end % if
    end % id
ic=ic+1;
end % while
%casticeeni pomoci bodu kinetochory
ix=1:2:keep;
xp=ceil(rand(1,M)*(Nt-1));
pop(keep+ix,:)=[pop(ma,1:xp) pop(pa,xp+1:Nt)];
pop(keep+ix+1,:)=[pop(pa,1:xp) pop(ma,xp+1:Nt)];
%Mutace
nmut=ceil((velikost_popuplace-1)*Nt*mutrate);
mradek=ceil(rand(1,nmut)*(velikost_popuplace-1))+1;
msloup=ceil(rand(1,nmut)*Nt);
for ii=1:nmut
    pop(mradek(ii),msloup(ii))=abs(pop(mradek(ii),msloup(ii))-1);
end
castice(2:velikost_popuplace,:)=bin2normal(pop(2:velikost_popuplace,:),pocBits(1),pocBits(2),p
ocBits(3),pocBits(4),pocBits(5));
hodn_ucel(2:velikost_popuplace)=feval(ff,castice(2:velikost_popuplace,:));
%Serazeni populace
[hodn_ucel,ind]=sort(hodn_ucel);
castice=castice(ind,:); pop=pop(ind,:);
%Statistika
minc(poc_iteraci+1)=min(hodn_ucel);
meanc(poc_iteraci+1)=mean(hodn_ucel);
if poc_iteraci>15
    if minc(poc_iteraci-15)==minc(poc_iteraci)
        konecne_res(por,:)=[minc(poc_iteraci) poc_iteraci castice(1,1) castice(1,2)
castice(1,3) castice(1,4) castice(1,5)];
        break
    end
end
%Kriteria zastaceni
if poc_iteraci>maxit
```

```
    konecne_res(por,:)=[minc(poc_iteraci) poc_iteraci castice(1,1) castice(1,2) castice(1,3)
castice(1,4) castice(1,5)]
    break
end
[poc_iteraci hodn_ucel(1)];
toc
end
%Zobrazeni vysledku
figure(1);
plot(minc)
hold on
plot(length(minc),minc(end),'ro')
title('CGAM fce; Pop. 300' )
xlabel('Pocet generaci binarniho GA')
ylabel('Hodnota ucelove funkce')
hold on
hold on
end
```

Příloha 4 – Spojitý genetický algoritmus

```
% Spojity genetický algoritmus
%   * minimalizuje objektive funkci uvedenou jako ucelfce
%   * pred zacatkem se musi nastavit parametry ve vsech castech
%% Algoritmus byl vytvoren na zaklade literatury uvedene v bibliografii a za pomoci studii


n=100;
pr=100; nejresp=zeros(100,2);
for por=1:n
clear minc
tic
format long
%% zakladni nastaveni
ucelfce='ft_ackley';
ncastice=5;
varhi=9; varlo=-1;
maxit=200;
%%  parametry
vel_populace=50;
mut_pomer=.1;
selekce=0.5;
Nt=ncastice;
preziti=floor(selekce*vel_populace);
nmut=ceil((vel_populace-1)*Nt*mut_pomer);
M=ceil((vel_populace-preziti)/2);
%% pocatecni populace
poc_iter=0;
castice=((varhi-varlo)*rand(vel_populace,ncastice)+varlo);
cost=feval(ucelfce,castice);
[cost,ind]=sort(cost);
castice=castice(ind,:);
minc=min(cost);
meanc=mean(cost);
%%%%%%%%%%%%%%%%%%pro graf
figure;
mesh(X,Y,y); view(270,270); hold on
plot(castice(:,1),castice(:,2),'ko');
%% iteracni proces
while poc_iter<maxit
poc_iter=poc_iter+1;
%% Parovani
M=ceil((vel_populace-preziti)/2);
prob=flipud([1:preziti]'/sum([1:preziti]));
pravdeps=[0 cumsum(prob(1:preziti))'];
vyber1=rand(1,M);vyber2=rand(1,M);
ic=1;
while ic<=M
for id=2:preziti+1
    if vyber1(ic)<=pravdeps(id) && vyber1(ic)>pravdeps(id-1);ma(ic)=id-1;
end
    if vyber2(ic)<=pravdeps(id) && vyber2(ic)>pravdeps(id-1);pa(ic)=id-1;
end
end
ic=ic+1;
end
%% single point crossover
ix=1:2:preziti;
xp=ceil(rand(1,M)*Nt);
r=rand(1,M);
for ic=1:M
xy=castice(ma(ic),xp(ic))-castice(pa(ic),xp(ic));
castice(preziti+ix(ic),:)=castice(ma(ic),:);
castice(preziti+ix(ic)+1,:)=castice(pa(ic),:);
castice(preziti+ix(ic),xp(ic))=castice(ma(ic),xp(ic))-r(ic).*xy;   %1st
castice(preziti+ix(ic)+1,xp(ic))=castice(pa(ic),xp(ic))+r(ic).*xy; %2nd
if xp(ic)<ncastice
castice(preziti+ix(ic),:)=[castice(preziti+ix(ic),1:xp(ic)),castice(preziti+ix(ic)+1,xp(ic)+1:
ncastice)];
castice(preziti+ix(ic)+1,:)=[castice(preziti+ix(ic)+1,1:xp(ic)),
castice(preziti+ix(ic),xp(ic)+1:ncastice)];
end         %if
end
%%  mutace
mradek=sort(ceil(rand(1,nmut)*(vel_populace-1))+1);
```

```
msloup=ceil(rand(1,nmut)*Nt);
for ii=1:nmut
castice(mradek(ii),msloup(ii))=(varhi-varlo)*rand+varlo;    % mutace
end % ii
%% evaluace
cost=feval(ucelfce,castice);
%% razeni
[cost,ind]=sort(cost);
castice=castice(ind,:);
%% statistika
minc(poc_iter+1)=min(cost);
meanc(poc_iter+1)=mean(cost);
if poc_iter>15
    if minc(poc_iter-15)==minc(poc_iter)
        konecne_res(por,:)=[minc(poc_iter) poc_iter castice(1,1) castice(1,2) castice(1,3)
castice(1,4) castice(1,5)];
        break
    end
end
nejres(por,:)=[castice(1,:)];
%%%%%%%%%%%%%%%%%%%%%%%%%pro graf
figure;
mesh(X,Y,y); view(270,270); hold on
plot(castice(:,1),castice(:,2),'ko');
%% stop
if poc_iter>maxit || cost(1)<mincost
    konecne_res(por,:)=[minc(poc_iter) poc_iter castice(1,1) castice(1,2) castice(1,3)
castice(1,4) castice(1,5)]
    break
end
[poc_iter cost(1)];
end %poc_iter
%%  Vysledek
castice(1,:)
figure(1);
plot(minc)
title('CGAM funkce; Pop. 300' )
xlabel('Pocet generaci')
ylabel('Hodnota ucelove funkce')
hold on
hold on
plot(length(minc),minc(end),'ro')
end
```

Příloha 5 – Optimalizace hejnem

```
% Optimalizace hejnem castic
%% Algoritmus byl vytvoren na zaklade literatury uvedene v bibliografii a za pomoci studii

clear all
for repeat=1:100
%%
ff = 'ft_ackley';
clear minc meanc realmin
velikost_populace=25;
pocet_castic=5;
maxit=200;
c1=1;
c2=4-c1;
C=.3;
varhi=9;
varlo=-1;
castice=(varhi-varlo)*rand(velikost_populace,pocet_castic)+varlo;
vel= rand(velikost_populace,pocet_castic);
hodn_ucel=feval(ff,castice);
minc=min(hodn_ucel);
meanc=mean(hodn_ucel);
globalmin=minc;
localcastice=castice;
localhodn_ucel=hodn_ucel;
[globalhodn_ucel,indx] = min(hodn_ucel);
globalcastice=castice(indx,:);
%%pro graf
[X,Y] = meshgrid(varlo:0.1:varhi);
y=ft_ackley2(X,Y);
figure;
mesh(X,Y,y); view(270,270); hold on
plot(castice(:,1),castice(:,2),'ko');
%START ITERACI
iter=0;
while iter < maxit
    iter=iter+1;
    w=(maxit-iter)/maxit;
    r1=rand(velikost_populace,pocet_castic); r2=rand(velikost_populace,pocet_castic);
    if length(globalcastice)<velikost_populace
    globalcastice=repmat(globalcastice,velikost_populace,1);
     end
    vel = C*(w*vel+c1*r1.*(localcastice-castice)+c2*r2.*(globalcastice-castice));
    castice= castice+vel; overlimit=castice<=varhi; underlimit=castice>=varlo;
    castice=castice.*overlimit+not(overlimit)*varhi;
    castice=castice.*underlimit+not(underlimit)*varlo;
hodn_ucel = feval(ff,castice);
betterhodn_ucel = hodn_ucel < localhodn_ucel;
localhodn_ucel = localhodn_ucel.*not(betterhodn_ucel) + hodn_ucel.*betterhodn_ucel;
localcastice(betterhodn_ucel) = castice(betterhodn_ucel);
[temp, t] = min(localhodn_ucel);
if temp<globalhodn_ucel
globalcastice=castice(t,:); indx=t; globalhodn_ucel=temp;
end
minimum=min(hodn_ucel);
meanimum=mean(hodn_ucel);
minc(iter+1)=minimum;
globalmin(iter+1)=globalhodn_ucel;
meanc(iter+1)=meanimum;
realmin(iter+1)=min(minc(:));
%%pro graf
[X,Y] = meshgrid(varlo:0.1:varhi);
y=ft_ackley2(X,Y);
figure;
mesh(X,Y,y); view(270,270); hold on
plot(castice(:,1),castice(:,2),'ko');
if iter>15
    if realmin(iter-15)==realmin(iter)
        konecne_res(repeat,:)=[realmin(iter) iter castice(1,1) castice(1,2) castice(1,3)
castice(1,4) castice(1,5)];
        break
    end
end
end                         %
```

```
konecne_res(repeat,:)=[realmin(iter) iter castice(1,1) castice(1,2) castice(1,3) castice(1,4)
castice(1,5)];
nej=(globalcastice(1,:));
globalnireseni1(repeat,:)=nej;
disp('øešení je');
repeat
globalnireseni=num2str(globalcastice(1,:))
figure(24)
hold on
plot(realmin(2:end),'b');
plot(length(realmin(2:end)),realmin(end),'ro')
title('CGAM funkce; Populace = 300; Pocet mereni = 100' )
xlabel('Pocet generaci')
ylabel('Hodnota ucelove funkce')
hold on
hold on
end
```

Příloha 6 – Hybridní GA-PSO algoritmus

```
% HYBRID GA-PSO ALGO
clear all
n=100;
pr=100; nejresp=zeros(100,2);
for pocet_mereni=1:n
clear realmin minc
tic
format long
%% zakladni nastaveni
ucelova_funkce='optimal_fceC';
pocet_castic=5;
max_hranice=1000; min_hranice=0;
ga_konec=7;
pso_konec=8;
maxit=300;
%%  castice parametry
velikost_populace=300;
mut_pomer=.1;
selekce=0.5;
Nt=pocet_castic;
keep=floor(selekce*velikost_populace);
nmut=ceil((velikost_populace-1)*Nt*mut_pomer);
M=ceil((velikost_populace-keep)/2);
pocet_poc_iter=0;                              castice=((max_hranice-
min_hranice)*rand(velikost_populace,pocet_castic)+min_hranicecost=feval(ucelova_funkce,castice
);
[cost,ind]=sort(cost);
castice=castice(ind,:);
realmin=min(cost);
meanc=mean(cost);
%% poc_iteracni proces
while pocet_poc_iter<maxit
pocet_poc_iter=pocet_poc_iter+1;
M=ceil((velikost_populace-keep)/2);
prob=flipud([1:keep]'/sum([1:keep]));
pravdeps=[0 cumsum(prob(1:keep))'];
vyber1=rand(1,M);  vyber2=rand(1,M);
ic=1;
while ic<=M
for id=2:keep+1
if vyber1(ic)<=pravdeps(id) && vyber1(ic)>pravdeps(id-1)
ma(ic)=id-1;
end
if vyber2(ic)<=pravdeps(id) && vyber2(ic)>pravdeps(id-1)
pa(ic)=id-1;
end
end
ic=ic+1;
end
%% krizeni na zaklade jednoho bodu kinetochory
ix=1:2:keep;
xp=ceil(rand(1,M)*Nt);
r=rand(1,M);
for ic=1:M
xy=castice(ma(ic),xp(ic))-castice(pa(ic),xp(ic));
castice(keep+ix(ic),:)=castice(ma(ic),:);
castice(keep+ix(ic)+1,:)=castice(pa(ic),:);
castice(keep+ix(ic),xp(ic))=castice(ma(ic),xp(ic))-r(ic).*xy;
castice(keep+ix(ic)+1,xp(ic))=castice(pa(ic),xp(ic))+r(ic).*xy;
if xp(ic)<pocet_castic
castice(keep+ix(ic),:)=[castice(keep+ix(ic),1:xp(ic)),castice(keep+ix(ic)+1,xp(ic)+1:pocet_cas
tic)];
castice(keep+ix(ic)+1,:)=[castice(keep+ix(ic)+1,1:xp(ic)),
castice(keep+ix(ic),xp(ic)+1:pocet_castic)];
end
end
%%  mutace
mrow=sort(ceil(rand(1,nmut)*(velikost_populace-1))+1);
mcol=ceil(rand(1,nmut)*Nt);
for ii=1:nmut
castice(mrow(ii),mcol(ii))=(max_hranice-min_hranice)*rand+min_hranice;    % mutace
end % ii
%% evaluace
cost=feval(ucelova_funkce,castice);
```

```
%% sortovani
[cost,ind]=sort(cost);
castice=castice(ind,:);
%% statistika
realmin(pocet_poc_iter+1)=min(cost);
meanc(pocet_poc_iter+1)=mean(cost);


%% stop kriteria
if pocet_poc_iter>maxit || cost(1)<mincost
    %konecne_res(por,:)=[minc(pocet_poc_iter) pocet_poc_iter castice(1,1) castice(1,2)]
    break
end
if pocet_poc_iter>ga_konec
    if realmin(pocet_poc_iter-ga_konec)<realmin(pocet_poc_iter)+0.001
        zastaveni_GA(pocet_mereni,:)=[realmin(pocet_poc_iter+1) pocet_poc_iter+1 castice(1,1)
castice(1,2)];
        break
    end
end
end
%%%% PRATICLE SWARM CAST
maxit=200;
 c1=1;
c2=4-c1;
C=0.5;
vel= rand(velikost_populace,pocet_castic);
cost=feval(ucelova_funkce,castice);


minc=min(cost);
meanc=mean(cost);
globalmin=minc;
localcastice=castice;
localcost=cost;
[globalcost,indx] = min(cost);
globalcastice=castice(indx,:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%START poc_iterACI
poc_iter=0;
while pocet_poc_iter < maxit
    pocet_poc_iter=pocet_poc_iter+1;
    poc_iter=poc_iter+1;
    w=(maxit-poc_iter)/maxit;
    r1=rand(velikost_populace,pocet_castic);    r2=rand(velikost_populace,pocet_castic);
    if length(globalcastice)<velikost_populace
       globalcastice=repmat(globalcastice,velikost_populace,1);
     end
    vel = C*(w*vel+c1*r1.*(localcastice-castice)+c2*r2.*(globalcastice-castice));
    castice= castice+vel;
    nadlimit=castice<=max_hranice;
    podlimit=castice>=min_hranice;
    castice=castice.*nadlimit+not(nadlimit)*max_hranice;
    castice=castice.*podlimit+not(podlimit)*min_hranice;
    cost = feval(ucelova_funkce,castice);
bettercost = cost < localcost;
localcost = localcost.*not(bettercost) + cost.*bettercost;
localcastice(bettercost) = castice(bettercost);
[temp, t] = min(localcost);
if temp<globalcost
globalcastice=castice(t,:); indx=t; globalcost=temp;
end
minimum=min(cost);
meanimum=mean(cost);
minc(poc_iter+1)=minimum;
globalmin(pocet_poc_iter+1)=globalcost;
meanc(pocet_poc_iter+1)=meanimum;
a=[min(minc) min(realmin)];
realmin(pocet_poc_iter+1)=min(a);
if poc_iter>pso_konec
    if realmin(pocet_poc_iter-pso_konec)<realmin(pocet_poc_iter)+0.001
        konecne_res(pocet_mereni,:)=[realmin(pocet_poc_iter+1) pocet_poc_iter+1 castice(1,1)
castice(1,2) castice(1,3) castice(1,4) castice(1,5)];
        break
    end
end
```

```
end
konecne_res(pocet_mereni,:)=[realmin(pocet_poc_iter+1) pocet_poc_iter+1 castice(1,1)
castice(1,2) castice(1,3) castice(1,4) castice(1,5)];
nej=(globalcastice(1,:));
globalnireseni=num2str(globalcastice(1,:));
figure(24)
hold on
plot(realmin(1:end),'b');
plot(length(realmin),realmin(end),'ro')
plot(realmin(1:zastaveni_GA(pocet_mereni,2)),'g'), axis tight
title('CGAM funkce; Pop. 300' )
xlabel('Pocet generaci hybridniho GA-PSO')
ylabel('Hodnota ucelove funkce')
hold on
hold on
end
```

Příloha 7 – Hybridní PSO-GA algoritmus

```
% HYBRIDNI PSO GA ALGORITMUS
clear all
repeat =1;por=1;
for repeat=1:100
%%
ucelova_funkce = 'optimal_fceC';
clear minc meanc realmin par cost
por=repeat;
velikost_populace=300;     pocet_castic=5;         maxit=200;
c1=1;
c2=4-c1;
C=0.5;
varhi=1000;
varlo=0;
par=((varhi-varlo)*rand(velikost_populace,pocet_castic)+varlo);
vel= rand(velikost_populace,pocet_castic);
cost=feval(ucelova_funkce,par);
minc=min(cost);
meanc=mean(cost);
globalni_minimum=minc;
realmin=min(cost);
localpar=par;
localcost=cost;
[globalcost,indx] = min(cost);
globalpar=par(indx,:);
%START ITERACI
iter=0;
while iter < maxit
    iter=iter+1;
    w=(maxit-iter)/maxit;
    r1=rand(velikost_populace,pocet_castic);
    r2=rand(velikost_populace,pocet_castic);
    if length(globalpar)<velikost_populace
       globalpar=repmat(globalpar,velikost_populace,1);
     end
    vel = C*(w*vel+c1*r1.*(localpar-par)+c2*r2.*(globalpar-par));
    par= par+vel;
    overlimit=par<=varhi;
    underlimit=par>=varlo;
    par=par.*overlimit+not(overlimit)*varhi;
    par=par.*underlimit+not(underlimit)*varlo;
    cost = feval(ucelova_funkce,par);
bettercost = cost < localcost;
localcost = localcost.*not(bettercost) + cost.*bettercost;
localpar(bettercost) = par(bettercost);
 [temp, t] = min(localcost);
if temp<globalcost
globalpar=par(t,:); indx=t; globalcost=temp;
end
minimum=min(cost);
meanimum=mean(cost);
minc(iter+1)=minimum;
globalni_minimum(iter+1)=globalcost;
meanc(iter+1)=meanimum;
realmin(iter+1)=min(minc(:));
if iter>8
    if realmin(iter-7)<realmin(iter)+0.001
        konecne_res(repeat,:)=[realmin(iter) iter par(1,1) par(1,2) par(1,3) par(1,4)
par(1,5)];
        break
    end
end
end                          %
konecne_res(repeat,:)=[realmin(iter) iter par(1,1) par(1,2) par(1,3) par(1,4) par(1,5)];
nej=(globalpar(1,:));
globalnireseni1(repeat,:)=nej;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mutrate=.1;selection=0.5;
Nt=pocet_castic;
popsize=Nt;
keep=floor(selection*velikost_populace);
nmut=ceil((popsize-1)*Nt*mutrate);
M=ceil((popsize-keep)/2);
iga=0;
```

```
cost=feval(ucelova_funkce,par);
[cost,ind]=sort(cost);
par=par(ind,:);
minc(iter+iga)=min(cost);
meanc=mean(cost);
realmin(iter+iga)=min(minc(:));
%% iteracni proces
while iga<maxit
iga=iga+1;
%% pairing
M=ceil((popsize-keep)/2);       prob=flipud([1:keep]'/sum([1:keep]));
odds=[0 cumsum(prob(1:keep))'];
pick1=rand(1,M);                pick2=rand(1,M);                ic=1;
while ic<=M
for id=2:keep+1
if pick1(ic)<=odds(id) && pick1(ic)>odds(id-1)
ma(ic)=id-1;
end
if pick2(ic)<=odds(id) && pick2(ic)>odds(id-1)
pa(ic)=id-1;
end
end
ic=ic+1;
end
ix=1:2:keep;
xp=ceil(rand(1,M)*Nt);
r=rand(1,M);
for ic=1:M
xy=par(ma(ic),xp(ic))-par(pa(ic),xp(ic));
par(keep+ix(ic),:)=par(ma(ic),:);
par(keep+ix(ic)+1,:)=par(pa(ic),:);
par(keep+ix(ic),xp(ic))=par(ma(ic),xp(ic))-r(ic).*xy;   %
par(keep+ix(ic)+1,xp(ic))=par(pa(ic),xp(ic))+r(ic).*xy; %
if xp(ic)<pocet_castic
par(keep+ix(ic),:)=[par(keep+ix(ic),1:xp(ic)),par(keep+ix(ic)+1,xp(ic)+1:pocet_castic)];
par(keep+ix(ic)+1,:)=[par(keep+ix(ic)+1,1:xp(ic)), par(keep+ix(ic),xp(ic)+1:pocet_castic)];
end         %if
end
%%  mutace
mrow=sort(ceil(rand(1,nmut)*(popsize-1))+1);
mcol=ceil(rand(1,nmut)*Nt);
for ii=1:nmut
par(mrow(ii),mcol(ii))=(varhi-varlo)*rand+varlo;
end % ii
%% evaluace
cost=feval(ucelova_funkce,par);
%% sortovani
[cost,ind]=sort(cost);
par=par(ind,:);
%% statistika
minc(iga+iter+1)=min(minimum);
realmin(iga+iter+1)=min(minc(:));
meanc(iga+iter+1)=mean(cost);
if iga>7
    if minc(iga+iter-7)<minc(iga+iter)+0.001
        konecne_res(por,:)=[minc(iga+iter+1) iga+iter+1 par(1,1) par(1,2) par(1,3) par(1,4)
par(1,5)];
        break
    end
end
nejres(por,:)=[par(1,:)];
%% stop kriteria
if iga>maxit
    konecne_res(por,:)=[minc(iga) iga par(1,1) par(1,2) par(1,3) par(1,4) par(1,5)];
    break
end
[iga cost(1)];
end %iga
figure(1);
plot(realmin,'g')
hold on
plot(length(realmin),realmin(end),'ro')
plot(realmin(1:iter),'b');
%plot(meanc,'g')
title('Rosenbrockovo sedlo; Pop. 300' )
xlabel('Pocet generaci hybridniho PSO-GA')
ylabel('Hodnota ucelove funkce')
```

```
hold on
hold on
end
```